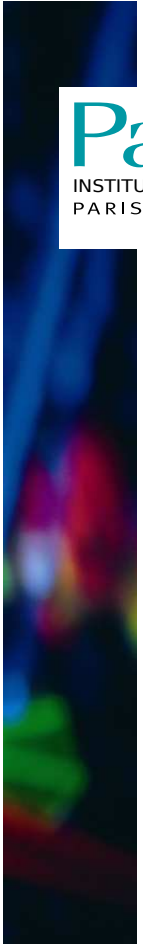
L'alignement de graphes : applications en bioinformatique et vision par ordinateur

Mikhail Zaslavskiy



ParisTech
INSTITUT DES SCIENCES ET TECHNOLOGIES
PARIS INSTITUTE OF TECHNOLOGY

MINES ParisTech

École doctorale n°431 : Information, Communication, Modélisation et Simulation

Doctorat ParisTech

T H È S E

pour obtenir le grade de docteur délivré par

l'École nationale supérieure des mines de Paris

Spécialité "Bioinformatique"

présentée et soutenue publiquement par

Mikhail Zaslavskiy

le 11 Janvier 2010

Graph matching and its application in computer vision and bioinformatics

Directeurs de thèse: Francis BACH and Jean-Philippe VERT

Jury:

Colin DE LA HIGUERA	Président
Martial HEBERT	rapporteur
Benno SCHWIKOWSKI	rapporteur
Nicola CANCEDDA	examinateur
Francis BACH	examinateur
Jean-Philippe VERT	examinateur

MINES ParisTech
CBIO & CMM

35 rue Saint-Honoré, Fontainebleau

THÈSE

Résumé

Le problème d'alignement de graphes, qui joue un rôle central dans différents domaines de la reconnaissance de formes, est l'un des plus grands défis dans le traitement de graphes. Nous proposons une méthode approximative pour l'alignement de graphes étiquetés et pondérés, basée sur la programmation convexe concave. Une application importante du problème d'alignement de graphes est l'alignement de réseaux d'interactions de protéines, qui joue un rôle central pour la recherche de voies de signalisation conservées dans l'évolution, de complexes protéiques conservés entre les espèces, et pour l'identification d'orthologues fonctionnels. Nous reformulons le problème d'alignement de réseaux d'interactions comme un problème d'alignement de graphes, et étudions comment les algorithmes existants d'alignement de graphes peuvent être utilisés pour le résoudre.

Dans la formulation classique de problème d'alignement de graphes, seules les correspondances bijectives entre les noeuds de deux graphes sont considérées. Dans beaucoup d'applications, cependant, il est plus intéressant de considérer les correspondances entre des ensembles de noeuds. Nous proposons une nouvelle formulation de ce problème comme un problème d'optimisation discret, ainsi qu'un algorithme approximatif basé sur une relaxation continue.

Nous présentons également deux résultats indépendents dans les domaines de la traduction automatique statistique et de la bio-informatique. Nous montrons d'une part comment le problème de la traduction statistique basé sur les phrases peut être reformulé comme un problème du voyageur de commerce. Nous proposons d'autre part une nouvelle mesure de similarité entre les sites de fixation de protéines, basée sur la comparaison 3D de nuages atomiques.

Abstract

The graph matching problem is among the most important challenges of graph processing, and plays a central role in various fields of pattern recognition. We propose an approximate method for labeled weighted graph matching, based on a convex-concave programming approach which can be applied to the matching of large sized graphs. This method allows to easily integrate information on graph label similarities into the optimization problem, and therefore to perform labeled weighted graph matching. One of the interesting applications of the graph matching problem is the alignment of protein-protein interaction networks. This problem is important when investigating evolutionary conserved pathways or protein complexes across species, and to help in the identification of functional orthologs through the detection of conserved interactions. We reformulate PPI alignment as a graph matching problem, and study how state-of-the-art graph matching algorithms can be used for this purpose.

In the classical formulation of graph matching, only one-to-one correspondences are considered, which is not always appropriate. In many applications, it is more interesting to consider many-to-many correspondences between graph vertices. We propose a reformulation of the many-to-many graph matching problem as a discrete optimization problem and we propose an approximate algorithm based on a continuous relaxation.

In this thesis, we also present two interesting results in statistical machine translation and bioinformatics. We show how the phrase-based statistical machine translation decoding problem can be reformulated as a Traveling Salesman Problem. We also propose a new protein binding pocket similarity measure based on a comparison of 3D atom clouds.

Acknowledgments

First of all, I would like to thank my scientific advisors Jean-Philippe Vert and Francis Bach. I think that it was a great piece of luck that I met them four years ago. I admire their way of guiding Phd students, they somehow manage to keep an optimal ratio between student research freedom and the value of their research, making sure that students spend their time usefully. It was always very easy for me to get their advice and meet them in person, when I needed it. At the same time, they always kept an eye on me, directing my research efforts in the right direction and encouraging my work. I have very happy memories of the exchange of emails the nights before deadlines, of long discussions with Jean-Philippe, when he asked me sequences of questions and helped me to answer them in order to make me solve yet another problem "by myself", of brainstormings with Francis in front of a blackboard where he guided me in the construction of more and more interesting convex relaxations. Thank you very much for all you have done for me !

I would also like to express my gratitude to all other members (former and current) of the Center for Computational Biology, Pierre, Martial, Laurent, Brice, Veronique, Fantine, Christian, Yoshi, Kevin, Isabelle, Nathalie, Anne-Claire, Franck, Toby, Philippe for their help, for many interesting discussions and a lot of happy times ("le cbio c'est bien !").

I spent most of the time during my Phd at the campus of Mines ParisTech in Fontainbleau where I met a lot of good friends with whom I shared a lot of happy moments playing volleyball, football and waiting for the school bus, and I would like to thank them for this.

I would also like thank my colleagues from the laboratory U900 (Insitut Curie)

and from the Willow-team (ENS-INRIA) for many valuable discussions and their willingness to help me in my work.

During my Phd, I had the chance to spend three months at Xerox Research Center Europe in Grenoble, and I am profoundly grateful to Marc and Nicola for inviting me there and working with me. In spite of the very short time, we did an excellent job together. Many thanks to all other members of XRCE for a warm welcome, hiking in mountains and teaching me climbing.

My Phd defense would not be possible without my jury: Martial Hebert, Benno Schwikowski, Nicola Cancedda and Colin De La Higuera, thank you very much for being there and for your comments on my work.

Finally, I would like to thank my family for supporting me all these years, and especially Tanya for her tolerance towards my endless deadlines and keeping me always ready to do the research. The day before my Phd defense, she told me "I am so happy that you will finally get you Phd" (by the way, it reminds me some stories from phdcomics.org). Now, I have it.

Contents

List of Tables

List of Figures

Introduction

This thesis consists of three independent parts. In the first (main) part, we present our principal results related to the graph matching problem. The second part contains a new result in the field of statistical machine translation. Finally, in the third and last part, we present a new method for comparing protein binding pockets which can be used for ligand prediction. In this section, we provide a short introduction to the topics discussed in this thesis as well as a brief description of the results that have been obtained.

1 The graph matching problem

Nowadays, the application of graph-based representation techniques to pattern recognition and machine learning is becoming more and more popular. When we need to classify objects with complex internal structures, it is not always possible to construct feature vectors that capture important discriminative information between object classes. These difficulties lead to the use of more complex techniques, in particular, graph-based representation methods, where graphs are used to encode object features and structural relationships between them. Graphs provide a universal and flexible tool which may be used to describe objects in many application areas: computer vision, bioinformatics, chemoinformatics, etc.

The question of efficient graph-based representation is a problem in itself. Depending on the area of application, different principles are used. In some cases, for example in chemoinformatics, it is very easy to construct a graph-based representation of molecules. In other cases, for example in computer vision, this is more tricky,

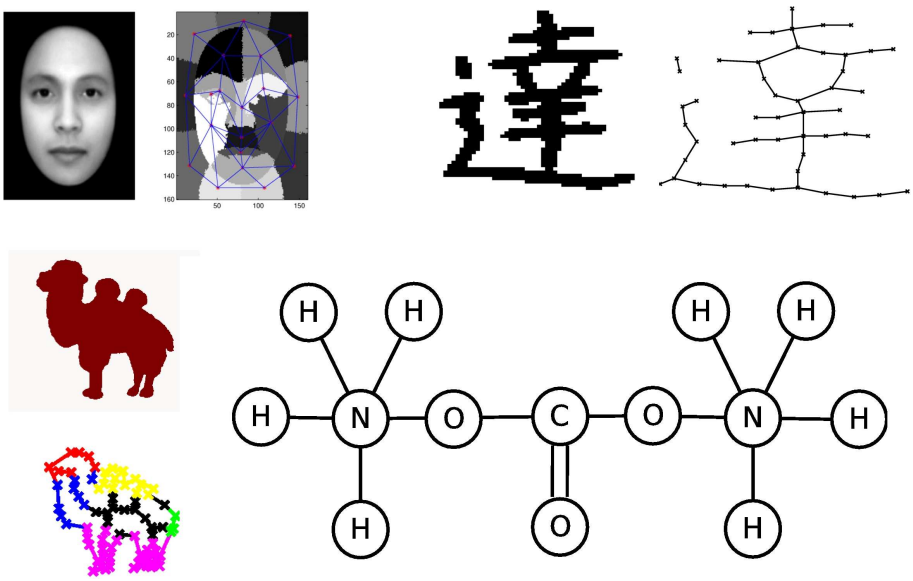


Figure 1: Examples of graph-based representations in computer vision and chemoinformatics.

there are several ways to represent the same image: we can use segmentation graphs, shock graphs, contour graphs or their combination. Here, we do not consider how objects can be represented by graphs, we suppose that a particular graph-based representation method has already been chosen and we are interested in what happens afterwards.

Once a graph representation has been constructed, a central question arising in the context of pattern recognition is the question of graph (dis-)similarity measure. To be able to classify or cluster objects on the basis of their graph-based representations, we need to know how to compare graphs. A natural method for graph comparison is based on graph alignment with further evaluation of alignment quality, the better the constructed alignment, the more similar the graphs. Construction of graph alignment is the subject of the graph matching problem where we seek a mapping between vertices of two graphs which optimally aligns the graph structures. The graph matching problem is among the most important challenges of graph processing, and plays a central role in various fields of pattern recognition. Usually, the optimality refers to the alignment of graph structures and, when available,

of vertex labels, although other criteria are also possible. A non-exhaustive list of graph matching applications includes document processing tasks like optical character recognition [Lee and Liu, 1999; Filatov et al., 1995], image analysis (2D and 3D) [Wang and Hancock, 2006; Luo and Hancock, 2000; Carcassoni and Hancock, 2003; Schellewald and Schnorr, 2005], and bioinformatics [Singh et al., 2008; Wang et al., 2004; Taylor, 2002]. Figure 1 gives some examples where graph matching can be used to compare graph-based representations of different objects such as images and molecules.

We formulate the graph matching problem as a least square optimization problem on the set of permutation matrices

$$\begin{aligned} &F(P) = ||A_G - PA_HP^T||_F^2 \\ &P \in \mathcal{P}, \end{aligned} \tag{1}$$

where A_G denotes the adjacency matrix of graph G, A_H denotes the adjacency matrix of graph H, and P denotes a permutation matrix. Here, for simplicity, we suppose that the graphs have the same number of vertices, the general case of graphs of different sizes is considered in Section 1.3.3. Adjacency matrices are square binary (or real-valued) matrices. The set of permutation matrices $\mathcal{P}$ is defined as a set of square binary matrices with only one non-zero element in each row and each column $\mathcal{P} = \{P \in \{0,1\}^N \times N : P1_N = 1_N,\ P^T1_N = 1_N\}$. We use permutation matrices to encode matchings between graphs, P_{ij} equals one if vertex i of graph G is aligned with vertex j of graph H. Function $F(P)$ in (1) represents the discrepancy between the graphs after matching P. If graphs G and H are simple unweighted graphs (with binary adjacency matrices), then $F(P)$ corresponds to the number of edges which are present in one graph but not in the other. In the case of weighted graphs, $F(P)$ represents the total difference between all overlapping edges.

Problem (1) is a difficult combinatorial problem (NP-hard in the general case). While some methods based on incomplete enumeration may be applied to search for an exact optimal solution in the case of small or sparse graphs, only approximate algorithms that usually find non-optimal solutions but are more scalable can be used

for large non-sparse graph matching. Many such approximate algorithms have been proposed, see e.g., the review by Conte et al. [2004]. Roughly speaking, there are three main categories of approximate algorithms.

The first group consists of approximate tree search algorithms [Bunke, 1983]. The general idea of these algorithms is quite simple, we construct the global mapping iteratively. First, we match the first vertex of graph G to a vertex of graph H, then at each step we match a new pair of vertices in order to maximize the current number of overlapping edges.

The second category represents spectral methods [Umeyama, 1988; Caelli and Kosinov, 2004; Leordeanu and Hebert, 2005; Cour et al., 2006; Leordeanu et al., 2007]. For example, Umeyama [1988]; Caelli and Kosinov [2004] use the spectral decomposition of graph adjacency matrices

$$A_G = V_G \Lambda_G V_G^T, \; A_H = V_H \Lambda_H V_H^T.$$

Rows of V_G and V_H can be seen as spectral coordinates of graph vertices, therefore to construct a matching between G and H, we match vertices with similar spectral coordinates.

The third category includes methods based on a relaxation of the optimization problem (1) [Almohamad and Duffuaa, 1993; Gold and Rangarajan, 1996].

Defining a similarity measure for graphs is not the only application where graph matching algorithms may be of great use. In classification or clustering problems we use graph matching as a similarity measure between objects of interest i.e the value of function $F(P)$, but we never use the optimal mapping itself. In some bioinformatics applications, the situation is quite the opposite, we are interested in the matrix P rather than in $F(P)$. An important example of such an application is the alignment of biological networks. For example, when we consider protein-protein interaction networks (see Figure 2), our objective is to find a mapping between proteins of two species which maximizes the number of conserved interactions. This problem is an instance of the graph matching problem where proteins correspond to graph

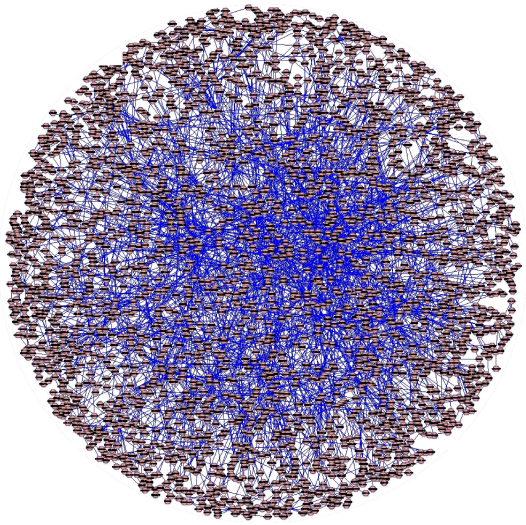


Figure 2: Fly protein-protein interaction network. Vertices (nodes) represent proteins and edges correspond to protein-protein interactions.

vertices, and protein-protein interactions correspond to graph edges. Once an alignment between protein-protein interaction networks is constructed, the matched pairs of proteins can be seen as "equivalent" proteins playing similar functional roles.

An important drawback of the existing formulation (1) is that it is based on a one-to-one correspondence between graphs. In many applications, it seems more natural to consider many-to-many mappings. For example, in computer vision, in some situations the same object may have different graph-based representations depending on noise, point of view and other factors. In such a case, we may need to match several vertices to one vertex, or groups of vertices to groups of vertices.

1.1 Contribution & Perspectives

In the present work, we propose a new graph matching algorithm based on convex-concave programming. The convex-concave programming formulation is obtained by rewriting the weighted graph matching problem as a least square problem on the set of permutation matrices and relaxing it to two different optimization problems: a quadratic convex and a quadratic concave optimization problem on the set of doubly

stochastic matrices. The concave relaxation has the same global minimum as the initial graph matching problem, but the search for its global minimum is still a hard combinatorial problem. We therefore construct an approximation of the concave problem solution by following a solution path of a convex-concave problem obtained by linear interpolation of the convex and concave formulations, starting from the convex relaxation. This method allows to easily integrate the information on graph label similarities into the optimization problem, and therefore to perform labeled weighted graph matching. A detailed description of this method is presented in Chapter 2.

The alignment of protein-protein interaction networks is the subject of several research papers. Bandyopadhyay et al. [2006] proposed to use a Markov random fields model, and Singh et al. [2008] introduced the IsoRank method inspired by the PageRank algorithm. In Chapter 3 we reformulate PPI alignment as a graph matching problem, and investigate how state-of-the-art graph matching algorithms can be used for that purpose. We differentiate between two alignment problems, depending on whether strict constraints on protein matches are given, based on sequence similarity, or whether the goal is instead to find an optimal compromise between sequence similarity and interaction conservation during alignment. We propose new methods for both cases, and assess their performance on the alignment of the yeast and fly PPI networks. The new methods consistently outperform state-of-the-art algorithms, retrieving in particular 78% more conserved interactions than IsoRank for a given level of sequence similarity.

To deal with the many-to-many graph matching problem, we can use several alternative approaches. Tree search algorithms can be easily generalized to the case of many-to-many matching. In the many-to-many case, the size of the optimization set is much larger, so when we match a new pair of vertices, one of them may be already matched to another vertex. Nevertheless, the core of the tree search algorithm for many-to-many matching is the same as in the one-to-one case. Spectral methods also have a natural generalization to the many-to-many case. Now, instead of matching pairs of vertices having similar spectral coordinates, we cluster all vertices on the basis of their spectral representations, then vertices from the same cluster are matched to

each other.

In Chapter 4 we show how the many-to-many graph matching problem can be reformulated as a least square optimization problem. To the best of our knowledge, this is the first attempt to give a compact formulation of the many-to-many graph matching problem and one of the advantages of this formulation is that it leads to a natural approximate algorithm based on a continuous relaxation. The new algorithm works significantly better than existing approaches based on tree search and spectral decomposition.

Concerning future perspectives, there are a lot of interesting things to be done. The PATH algorithm probably can be further improved by construction of tighter convex and concave relaxations. The current procedure for processing directed graphs is based on a transformation of directed graph matching to undirected graph matching by doubling the number of vertices, but it would be better to run the PATH algorithm directly on directed graphs.

Since graph matching methods show good performance in the alignment of protein-protein interaction networks, it would be worth testing them on other types of biological networks such as gene co-expression networks. Another interesting direction is the so called multi-matching problem where we seek a simultaneous alignment of several networks. In this case, the problem is formulated as follows. We have three or more graphs, for instance, G, H and B and our objective is to find an alignment of the graphs which minimize the total discrepancy between all triples of overlapping edges (g_{ij}, h_{ij} and b_{ij})

$$\text{discrepancy} = (g_{ij} - h_{ij})^2 + (g_{ij} - b_{ij})^2 + (b_{ij} - h_{ij})^2.$$

Finally, we can formulate the three-graph multi-matching problem in the following way

$$\begin{aligned} &\min_{P_H, P_B} ||G - P_H H P_H^T||_F^2 + ||G - P_B H P_B^T||_F^2 + ||P_B H P_B^T - P_H H P_H^T||_F^2 \\ &\text{subject to } P_B, P_H \in \mathcal{P} \end{aligned} \tag{2}$$

or in the general case with n graphs $G_1,\ldots,G_n$

$$\begin{aligned} &\min_{P_1,P_2,\ldots,P_n} \sum_{i,j} ||P_i G_i P_i^T - P_j G_j P_j^T||_F^2 \\ &\text{subject to } P_1 = I, \quad P_2,\ldots,P_n \in \mathcal{P}. \end{aligned} \tag{3}$$

The majority of graph matching algorithms can be generalized to the multi-matching case, and it seems that this generalization may be quite useful is such fields of application as bioinformatics (synchronized alignment of biological networks corresponding to several species like Human-Mouse-Fly) or computer vision (synchronized alignment of graphs representing the same object).

2 Phrase-based statistical machine translation

One of the most famous challenges in natural language processing (NLP) is how to teach computers to translate texts. The objective is to construct a computer algorithm which can translate sentences from a source language (for example, French) to a target language (for example, English). There are two major groups of methods for machine translation: rule-based systems and statistical machine translation (SMT) methods. Rule-based systems use linguistic rules defined by a human expert. SMT methods learn a translation model by themselves from a parallel bilingual text corpora (set of sentences in the source language and their translations in the target language). Also, there exist so called hybrid translation models where one tries to combine the best features of rule-based systems with the best of SMT models.

Ones of the most successful SMT systems are so-called Phrase-Based SMT models. They use aligned sequences of words, called biphrases, as building blocks for translations. Figure 3 presents an illustration of PB-SMT. Roughly speaking, first, we segment the source sentence into blocks, then we translate them and finally the translated blocks are aligned in order to construct a correct sentence according to the target language. Note that in practice, "segmentation", "phrase translation" and "alignment" are performed simultaneously and not step by step.

Pairs of phrases such as *(est un, is a)* and *(problème combinatoire, combinatorial*

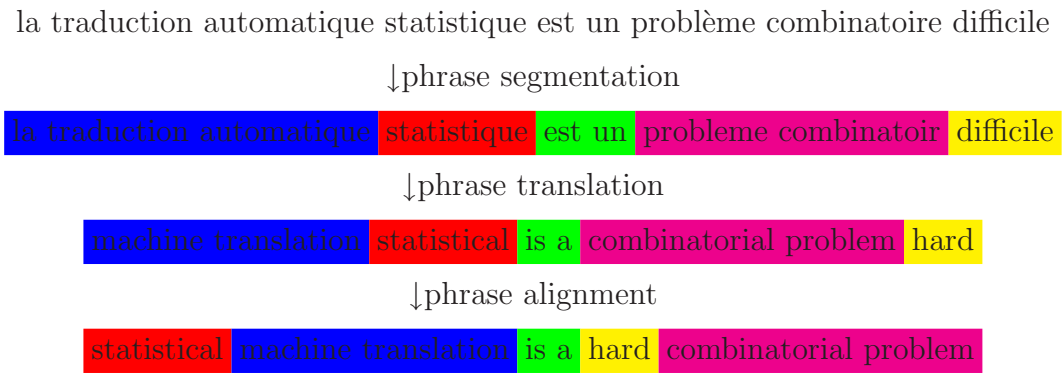


Figure 3: An example of phrase-based statistical machine translation process. Source and target parts of the same biphrases are highlighted by the same color.

problem) are called biphrases, so the translation process can be seen as a procedure where, first, we cover the source sentence by a set of biphrases and then we permute the selected biphrases in order to construct a plausible translation.

The entire translation, consisting of selected biphrases and a biphrase permutation is usually called *alignment* a. PB-SMT models score alternative candidate translations for the same source sentence based on a log-linear model of the conditional probability of translation given the source sentence

$$p(T, a|S) = \frac{1}{Z_S} \exp \sum_k \lambda_k h_k(S, a, T)\,, \tag{4}$$

where the h_k's are features, that is, functions of the source string S, of the target string T, and of the alignment a (a already contains T but to emphasize that our objective is to construct T, we write (T, a) instead of just a). The λ_k's are weights and Z_S is a normalization factor that guarantees that p is a proper conditional probability distribution over the pairs (T, a). During the training phase, we estimate all model parameters λ and construct a dictionary of biphrases, then to translate a new sentence S (inference phase) we have to solve the following optimization problem

$$(T^*, a^*) = \arg\max_{T,a} P(T, a|S). \tag{5}$$

Problem (5) is called *the Decoding problem.*

The majority of PB-SMT models use the following list of features (s_i and t_i denote the source and target components of biphrases)

Local features	Non-local features
Forward probabilities $p(\widetilde{t}_i\|\widetilde{s}_i)$	Language model $p(\widetilde{t}_i\|\widetilde{t}_{i-1},\ldots,\widetilde{t}_{i-n})$
Reverse probabilities $p(\widetilde{s}_i\|\widetilde{t}_i)$	Distortion $\|pos(\widetilde{s}_i) - pos(\widetilde{s}_{i-1})\|$
Phrase lengths $length(\widetilde{t}_i)$	

Local features depend only on individual biphrases, this means that they influence only the choice of biphrases, not their permutation. Non-local features depend on consecutive biphrases, they control both the permutation and choice of biphrases. If we use only local features in our translation model, then the decoding problem can be solved exactly in polynomial time, otherwise if we use non-local features such as the language model, then the decoding problem becomes NP-hard.

2.1 Contribution & Perspectives

Since, in the general case, the decoding problem is too hard to be solved exactly, approximate methods are normally used. The most used strategy for the decoding problem is the so-called beam-search algorithm which is a variant of the tree search strategy. In our joint work with Marc Dymetman and Nicola Cancedda from Xerox Research Center Europe (XRCE) we proposed an alternative decoding algorithm. We showed that the decoding problem is equivalent to the traveling salesman problem (TSP), a well known problem in operational research.

Given a non-directed graph G on N vertices, where the edges carry real-valued costs, the TSP problem consists in finding a tour of minimal total cost, where a tour (also called Hamiltonian Circuit) is a "circular" sequence of vertices visiting each vertex of the graph exactly once.

In Chapter 5, we propose a procedure which transforms any decoding problem into a traveling salesman problem. Given a new sentence S, this procedure constructs a graph where the optimal TSP tour corresponds to the solution of the decoding problem. Besides the general interest in this transformation for understanding decoding,

it also opens the door to the direct application of a variety of existing TSP algorithms to SMT. Our experiments on synthetic and real data show that fast TSP algorithms can handle selection and reordering in SMT comparably or better than the state-of-the-art beam-search strategy, converging on solutions a higher objective function in a shorter time.

For the moment, we use classical TSP algorithms, and one of the interesting future directions is to further improve the optimization strategy by taking into account the special structure of the decoding graph.

3 Comparison of protein binding pockets

One of the main goals of structural biology is to predict, from the 3D fold of a protein, its interacting partners, which in turn is related to its molecular function. Understanding this structure-function relationship is still an open question, and no reliable tool is available to make such a prediction. Current efforts concentrate on local 3D approaches, focusing on identification and comparison of binding pockets, in order to predict the natural ligand for a protein, with the underlying idea that proteins sharing similar binding sites are expected to bind similar ligands. The same strategy also applies to the problem of identifying new drug precursors for a therapeutic target protein.

Binding pockets may be seen as 3D cavities on the protein surface (see Figure 4), we are interested in a method which will be able to detect pockets binding the same ligand on the basis of their 3D structure. Given such a method, we will be able to predict binding ligands for new, previously, unseen proteins.

3.1 Contribution & Perspectives

In Chapter 6, we propose an approach in which binding pockets are represented by clouds of atoms in 3D space potentially baring additional labels such as partial charge or atom type. The new similarity measure is based on the alignment of protein pockets with the further use of a convolution kernel between 3D point clouds.

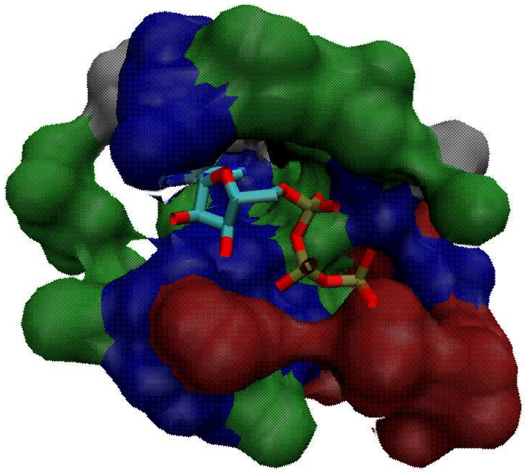


Figure 4: An illustration of an ATP binding pocket with the ATP ligand inside.

Let $P = (x_i, l_i)_{i=1}^N$ denote a binding pocket consisting of N atoms, where $x_i \in \mathbb{R}^3$ is a 3D vector representing atom coordinates, and l_i is a label (discrete or real valued) that may be used to bare additional information on the atoms (for example, atom type, atom partial charge, or amino acid type).

A classical approach for pocket comparison consists in iterative pocket alignment and further counting of overlapping atoms, usually within a tolerance of 1Å [Willett et al., 1986]. The alignment is made to maximize the number of overlapping atoms, which is generally a good indicator of pocket similarity.

However, atoms may have different positions but play equivalent roles in ligand binding, and the role of one atom in one pocket may be played by a group of atoms in another pocket. These observations lead us to the idea of an alternative smooth score which does not count the number of overlapping atoms, but rather uses a weighted number of atoms having closed positions. Given two pockets P_1 and P_2 the similarity measure $K(P_1, P_2)$ is defined as follows

$$K(P_1, P_2) = \sum_{x_i \in P_1} \sum_{y_j \in P_2} e^{\frac{-||x_i - y_j||^2}{2\sigma^2}}. \tag{6}$$

This similarity measure represents a positive definite kernel, σ characterizes the sensitivity of the similarity measure (6) to the relative displacements of atoms.

In practice, formula (6) is not fully appropriate, because the proposed measure is not invariant under rotations and translations of the binding pockets. Therefore, we define a similarity measure *sup-CK* as the maximum of (6) over all possible rotations and translations of one of the two pockets:

$$\text{sup-CK}(P_1, P_2) = \max_{R, y_t} \sum_{x_i \in P_1, y_j \in P_2} e^{\frac{||x_i - (Ry_j + y_t)||^2}{2\sigma^2}}, \tag{7}$$

This approach has shown good performance on several benchmark datasets in comparison with such methods as the Tanimoto index [Willett et al., 1986], the SitesBase algorithm [Gold and Jackson, 2006], the MultiBind algorithm [Shulman-Peleg et al., 2008] and a method based on real spherical harmonic expansion coefficients [Morris et al., 2005].

Regarding future research directions, it would be interesting to couple the proposed similarity measure with some similarity measure between ligands in order to further improve the prediction performance. Then such a combination may be a good basis for the development of a public web server for protein-ligand interaction prediction.

4 Publications

The results presented in this thesis were published in the following papers.

- Chapter 2: *M. Zaslavskiy, F. Bach, J-P. Vert* **A Path Following Algorithm for the Graph Matching Problem**, "IEEE Transactions on Pattern Analysis and Machine Intelligence", Dec. 2009.

- Chapter 3: *M. Zaslavskiy, F. Bach, J-P. Vert* **Global alignment of protein–protein interaction networks by graph matching methods** "Bioinformatics Oxford" (presented at ISMB-ECCB 2009).

- Chapter 5: *M. Zaslavskiy, M. Dymetman, N. Cancedda* **Phrase-Based Statistical Machine Translation as a Traveling Salesman Problem** "Proceedings of the 47th Annual Meeting of the Association for Computational Linguistics (ACL-IJCNLP 2009)", Jul. 2009.

- Chapter 6: *B. Hoffmann, M. Zaslavskiy , J-P. Vert, V. Stoven* **A new protein binding pocket similarity measure based on comparison of 3D atom clouds: application to ligand prediction**, conditionally accepted to BMC Bioinformatics, 2009

Part I

Graph matching

Chapter 1

Introduction and history of the graph matching problem

The goal of this chapter is to introduce the graph matching problem. We compare alternative formulations of graph matching and trace the evolution of ideas related to graph comparison.

1.1 Basic definitions and notations

- A **graph** $G = (V, E)$ of size N is defined by a finite set of vertices $V = \{1, \ldots, N\}$ and a set of edges $E \subset V \times V$. Each graph can be represented by a square **adjacency matrix** A of size $|V| \times |V|$, where A_{ij} is equal to one if there is an edge between vertex i and vertex j, and zero otherwise.

- In **weighted graphs**, edges have associated labels(weights). Weights are usually real numbers. Unweighted graphs are described by binary adjacency matrices.

- G is called an **undirected graph** if and only if $A^G_{ij} = A^G_{ji}$.

- $G' = (V', E')$ is a **subgraph** of graph G, if $V' \subset V$ and $E' \subseteq V' \times V' \cap E$. G' is called an **induced** subgraph of G if $E' = V' \times V' \cap E$. G'.

- A **matching** or an alignment of two graphs is a mapping between the vertices of two graphs

$$f : V^G \to V^H.$$

If graphs have the same number of vertices N and f is a bijection, then such a matching is called **one-to-one**. A one-to-one matching can be encoded by a **permutation matrix**. The set of permutation matrices is defined as follows

$$\mathcal{P} = \{P \in \{0,1\}^{N\times N} : P1_N = 1_N, P^T 1_N = 1_N\}, \tag{1.1}$$

where 1_N is a column vector with N ones.

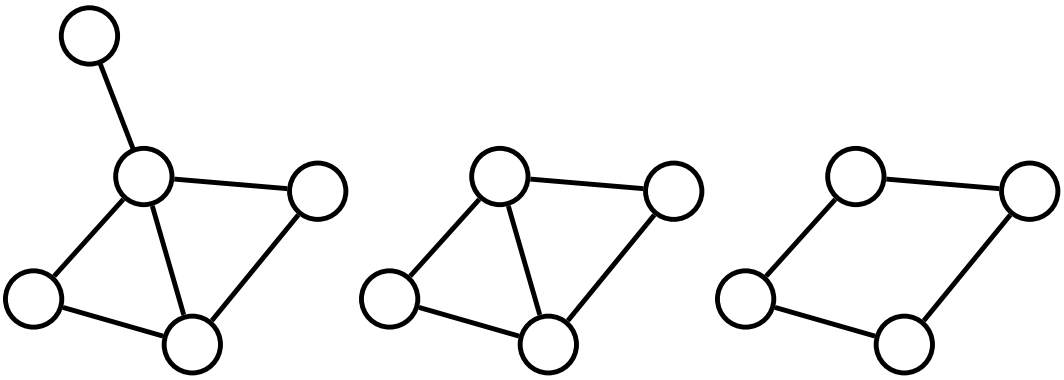
- Two graph G and H are called **isomorphic** if and only if there exists a one-to-one mapping $f : G \to H$ such that $(i,j) \in E^G \leftrightarrow (f(i), f(j)) \in E^H$

1.2 Formulation of the graph matching problem

The first formulation of the graph matching problem was proposed by Tsai and Fu [1979]. Graph matching was introduced as a noisy version of the graph isomorphism problem. Such a definition is quite natural for understanding the graph matching problem. Checking for graph isomorphism, we can only verify whether two graphs are the same or not, but in many applications, this is not enough. Sometimes even if graphs are different, we need to know how different they are, in other words, instead of a binary Yes/No answer for the graph isomorphism problem, we need a graph (dis-)similarity measure with more gradations. A possible solution is to use the size of the maximum common subgraph (MCS) as a measure of graph similarity, or its normalized version

$$\mathrm{Sim}(G,H) = \frac{|\mathrm{MCS}(G,H)|}{\max(|G|,|H|)}, \tag{1.2}$$

where $|G|$ denotes the number of edges in G. The classical definition of the maximum common subgraph is based on the notion of induced subgraphs. Figure 1.1 illustrates the difference between induced and simple subgraphs. Subgraph G' of graph G is called an induced subgraph if together with the selected vertices, it contains all edges



(a) graph G (b) induced subgraph (c) non-induced subgraph

Figure 1.1: Examples of induced and non-induced subgraphs.

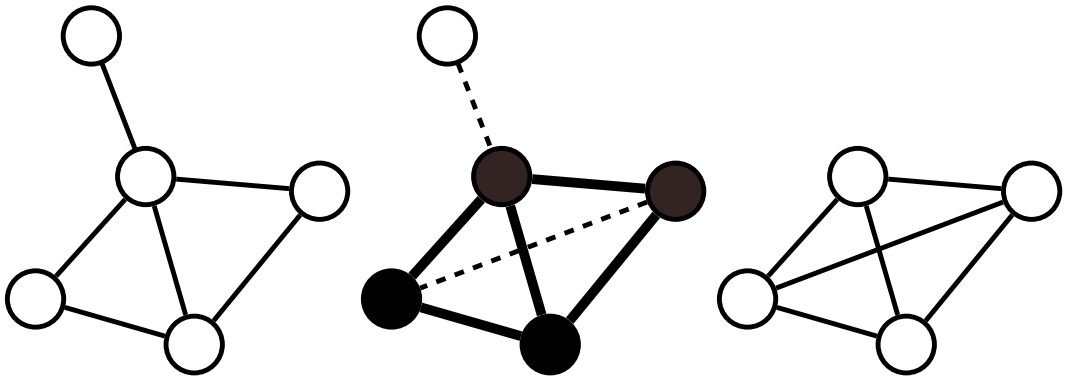
connecting these vertices.

Usually, the maximum common subgraph is defined as the maximum induced common subgraph, but in our case, to measure similarity (1.2) between graphs, we can use both versions: induced and non-induced.

However, this approach is appropriate only in the case of simple unweighted graphs. If graph edges have associated weights (which is often the case in real applications) then it becomes difficult to use the notion of maximum common subgraph. For instance, it is unreasonable to seek a common subgraph with exactly the same edge weights if these weights are real numbers. Of course, one can always discretize weights and follow the MCS approach, but this may lead to information loss.

To understand what would be a better alternative to the discretization schema, let us look at the maximum common subgraph problem from a different point of view. Let us suppose, that we do not consider only induced subgraphs, but all kinds of subgraphs. And let us suppose, for simplicity, that graphs G and H have the same number of vertices N. Then the extraction of the MCS may be seen as a procedure where we seek an alignment of two graphs which provides the maximum number of overlapping edges. This idea is illustrated in Figure 1.2.

The optimal alignment may be defined as an alignment maximizing the number of overlapping edges (solid lines) or minimizing the number of non-overlapping edges (dotted lines). Alignment of two graphs may be encoded by the permutation matrix



(a) graph G (b) MCS(G,H) (c) graph H

Figure 1.2: The maximum common subgraph (solid edges) as a result of graph alignment.

P where $P_{ij} = 1$ if vertex i of graph G is matched to vertex j of graph H and zero otherwise. Let G and H also denote the adjacency matrices of corresponding graphs, then the number of non-overlapping edges under matching P can be expressed as follows

$$F(P) = \frac{1}{2}||G - PHP^T||_F^2 \tag{1.3}$$

where $||\ ||_F$ denotes the Frobenius norm $||A||_F^2 = \sum A_{ij}^2$. Function $F(P)$ expresses the number of non-overlapping edges, and therefore the problem of MCS identification may be rewritten as the following optimization problem

$$\begin{aligned} &\min_P F(P) \\ &\text{subject to} \\ &P \in \{0,1\}^{N\times N}\,,\ P1_N = 1_N\,,\ P^T1_N = 1_N \end{aligned} \tag{1.4}$$

The choice of the Frobenius norm is quite arbitrary, it can be replaced by any matrix norm, for instance, l_p $(1 \leq p \leq \infty)$. The optimization set in (1.4) is exactly the set of permutation matrices. Now, given (1.3,1.4), the generalization to the case of weighted graphs is straightforward. The introduction of edge weights corresponds to replacing binary elements of matrices G and H by real numbers. Optimization problem (1.4) in the case of weighted graphs means that we seek a matching which minimize the

total difference between all aligned edges. Note, that if we consider weighted graphs, there is no longer any difference between induced and non-induced subgraphs, the absence of an edge between two vertices may be considered to be an edge with zero weight.

The formulation of graph matching in the form of (1.4) was given in [Umeyama, 1988], where Umeyama rewrote the idea of inexact graph isomorphism [Tsai and Fu, 1979] in the form of an optimization problem. We use this formulation, however there exist alternative formulations of the graph matching problem such as the graph edit distance. In the next section, we briefly discuss the relation between formulation (1.4) and other existing definitions of the graph matching problem.

1.3 Alternative formulations of the graph matching problem

In the general case, the graph matching problem is formulated as follows. Given two graphs, find the correspondence between their vertices which provides the best alignment of graph structures. This definition is informal since the notion of best alignment is not uniquely defined. Depending on how we define it, we get different formulations of the graph matching problem.

1. **Exact Matching**

    - Graph isomorphism: check whether two graphs are the same.
    - Subgraph isomorphism: check whether the smallest graph is a subgraph of the biggest one.

2. **Inexact Matching**

    - MCS: the maximum common subgraph problem.
    - Least square formulation: minimize (1.4).
    - Graph edit distance.

The variants of exact matching may be seen as particular cases of the least square formulation. If graphs G and H are of the same size, then they are isomorphic if and only if $\min_P F(P) = 0$. Similarly, if G is smaller than H, then G is a subgraph of H if and only if $\min_P F(P) = |H| - |G|$.

In the case of inexact matching, along with the least square formulation, another popular approach is based on the graph edit distance. The graph edit distance was proposed by Tsai and Fu [1979]; Bunke [1983]. It was defined as the minimum amount of distortion that we need to transform one graph into another. Graph transformation is performed via insertions, deletions and substitutions of graph vertices and edges. Each operation has an associated cost, and the transformation distortion is defined as the total cost of all operations employed. Interestingly, in many cases, the graph edit distance may be rewritten in terms of the least square formulation, we will show how this can be done in Section 1.3.5, but first we consider how the least square formulation may be further generalized to include information on vertex labels, how the graph matching problem may be rewritten in the form of a quadratic assignment problem and what can be done if graphs have a different number of vertices.

1.3.1 Vertex labels

An interesting instance of the graph matching problem is the matching of labeled graphs. In that case, graph vertices have associated labels, which may be numbers, categorical variables, etc... The important point is that there is also a similarity measure between these labels. Therefore, when we search for the optimal correspondence between vertices, we search a correspondence which matches not only the structures of the graphs but also vertices with similar labels. Some widely used approaches for this case only use the information about similarities between graph labels. In vision, one such algorithm is the shape context algorithm proposed by Belongie et al. [2002], which involves an efficient algorithm of node label construction. Another example is the BLAST-based algorithms in bioinformatics such as the Inparanoid algorithm [Brein et al., 2005], where correspondence between different protein networks is established on the basis of BLAST scores between pairs of proteins. The main advantages

of all these methods are their speed and simplicity. However, these methods do not take into account information about the graph structure. Some graph matching methods try to combine information on graph structures and vertex similarities, examples of such method are presented in [Schellewald et al., 2001; Singh et al., 2008].

The least square formulation may be easily adjusted to include information on the vertex labels. Let g_i and h_j denote vertex labels in graphs G and H correspondingly. The optimal alignment of two graphs should not only match edges with similar weights, but also put into correspondence vertices having similar labels. The new objective function is the following modification of (1.3)

$$F_\alpha(P) = (1-\alpha)||G - PHP^T||_F^2 + \alpha \mathrm{tr} CP^T\,, \tag{1.5}$$

where $C \in \mathcal{R}^{N\times N}$ encodes pairwise dissimilarities between vertex labels of two graphs $C_{ij} = \mathrm{dissim}(g_i, h_j)$, and α controls the trade-off between edge and vertex alignment components, the greater parameter α, the more attention we pay to alignment of vertices with similar labels.

1.3.2 Quadratic assignment problem

An interesting fact about the least square formulation is that it can be seen as an instance of the quadratic assignment problem. The quadratic assignment problem is formulated as follows

$$\begin{aligned} &\max_P \ \mathrm{tr} AP^T BP \\ &\text{subject to} \\ &P \in \{0,1\}^{N\times N}\,,\ P1_N = 1_N\,,\ P^T 1_N = 1_N\,, \end{aligned} \tag{1.6}$$

where A and B are $N \times N$ real valued matrices.

The transformation of (1.4) to (1.6) is quite simple, the optimization set is exactly the same (the set of permutation matrices) and we only need to rewrite the objective

function

$$\begin{aligned}||G - PHP^T||_F^2 &= ||GP - PHP||_F^2 \\ &= \mathrm{tr}P^TG^TGP - 2\mathrm{tr}P^TG^TPH + \mathrm{tr}H^TP^TPH \\ &= \mathrm{tr}G^TG - 2\mathrm{tr}G^TPHP^T + \mathrm{tr}H^TH\end{aligned}$$

now since $\mathrm{tr}G^TG$ and $\mathrm{tr}H^TH$ do not depend on P

$$\min_P ||G - PHP^T||_F^2 \Leftrightarrow \max_P \mathrm{tr}G^TPHP^T.$$

The information on vertex pairwise similarities (see the previous section) may be also included in the QAP formulation, it corresponds to adding a linear term to the QAP objective function. This extended formulation of QAP is usually called the generalized quadratic assignment problem.

1.3.3 Matching graphs of different sizes

Often in practice we have to match graphs of different sizes N_G and N_H ($N_G < N_H$). In this case we have to match all vertices of graph G to a subset of vertices of graph H. In the usual case when $N_G = N_H$, the error (1.3) corresponds to the number of mismatched edges (edges which exist in one graph and do not exist in the other one). When we match graphs of different sizes the situation is a bit more complicated. Let $V_H^+ \subset V_H$ denote the set of vertices of graph H that are selected for matching to vertices of graph G, let $V_H^- = V_H \setminus V_H^+$ denote all the rest. Therefore all edges of the graph H are divided into four parts $E_H = E_H^{++} \cup E_H^{+-} \cup E_H^{-+} \cup E_H^{--}$, where E_H^{++} are edges between vertices from V_H^+, E_H^{--} are edges between vertices from V_H^-, E_H^{+-} and E_H^{+-} are edges from V_H^+ to V_H^- and from V_H^- to V_H^+ respectively. For undirected graphs the sets E_H^{+-} and E_H^{+-} are the same (but, for directed graphs they would be different). The edges from E_H^{--}, E_H^{+-} and E_H^{-+} are always mismatched and a question is whether we have to take them into account in the objective function or not. According to the answer we have three types of matching error (four for directed graphs) with interesting interpretations.

1. We count only the number of mismatched edges between G and the chosen subgraph $H^+ \subset H$. It corresponds to the case when the matrix P from (1.3) is a matrix of size $N_G \times N_H$ and $N_H - N_G$ columns of P contain only zeros.

2. We count the number of mismatched edges between G and the chosen subgraph $H^+ \subset H$. And we also count all edges from E_H^{--}, E_H^{+-} and E_H^{-+}. In this case P from (2.1) is a matrix of size $N_H \times N_H$. And we transform A_G into a matrix of size $N_H \times N_H$ by adding $N_H - N_G$ zero rows and zero columns. It means that we add dummy isolated vertices to the smallest graph, and then we match graphs of the same size.

3. We count the number of mismatched edges between G and chosen subgraph $H^+ \subset H$. And we also count all edges from E_H^{+-} (or E_H^{-+}). It means that we count matching error between G and H^+ and we count also the number of edges which connect H^+ and H^-. In other words we are looking for subgraph H^+ which is similar to G and which is maximally isolated in the graph H.

Figure 1.3.3 illustrates different types of matching error described above.

Each type of error may be useful according to the context and interpretation, but a priori, it seems that the best choice is the second one where we add dummy nodes to the smallest graph. The main reason is the following. Suppose that graph G is quite sparse, and suppose that graph H has two candidate subgraphs H_s^+ (also quite sparse) and H_d^+ (dense). The upper bound for the matching error between G and H_s^+ is $\#V_G + \#V_{H_s^+}$, the lower bound for the matching error between G and H_d^+ is $\#V_{H_d^+} - \#V_G$. So if $\#V_G + \#V_{H_s^+} < \#V_{H_d^+} - \#V_G$ then we will always choose the graph H_s^+ with the first strategy, even if it is not similar at all to the graph G. The main explanation of this effect lies in the fact that the algorithm tries to minimize the number of mismatched edges, and not to maximize the number of well matched edges. In contrast, when we use dummy nodes, we do not have this problem because if we take a very sparse subgraph H^+ it increases the number of edges in H^-(the common number of edges in H^+ and H^- is constant and is equal to the number of edges in H) and finally it decreases the quality of matching.

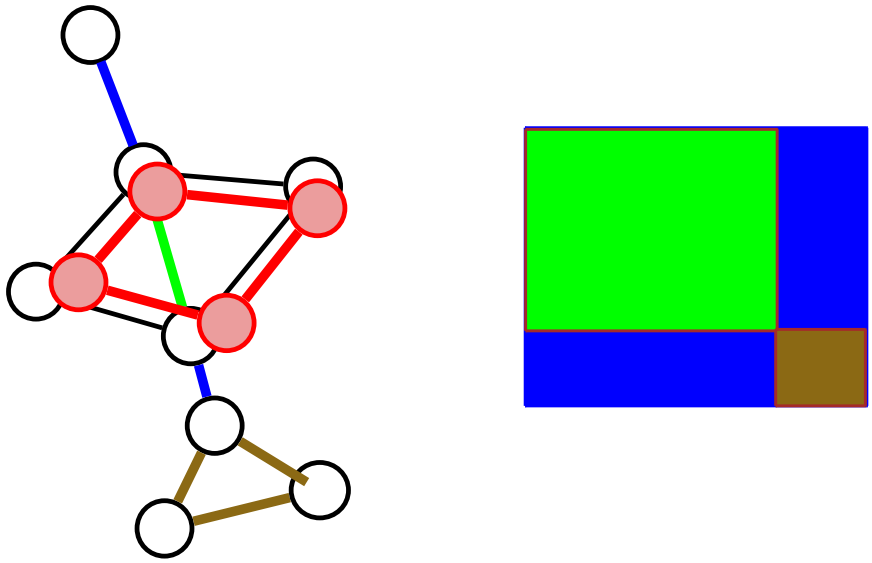


Figure 1.3: Matching of graphs with different number of vertices. On the left graph G is drawn in red, edge set E_H^{++} is drawn in black (matched edges) and green (mismatched edges), E_H^{+-} in blue and E_H^{--} in brown. On the right, the same edge sets are represented in terms of the adjacency matrices: green area corresponds to E_H^{++}, blue area to E_H^{+-} and brown area to E_H^{--}

1.3.4 l_1 and other alternatives to the l_2 norm in the graph matching problem

To solve the graph matching problem, we seek a mapping between the vertices of two graphs which minimizes the difference between the adjacency matrix of one graph and the permuted adjacency matrix of the second graph. In the least square formulation, the difference is measured by the l_2 norm, but, of course, any other matrix norm may be used. For example, the LP based approach [Almohamad and Duffuaa, 1993] uses the l_1 norm, other choices like l_p norms with different p are possible as well. Note, if we consider only unweighted graphs (with binary adjacency matrices), then all l_p norms (power p) are equivalent

$$||A||_1 = ||A||_2^2 = ||A||_p^p \quad \forall\ 1 \leq p < \infty$$

1.3.5 Graph edit distance

The graph edit distance, which has already been defined above, is based on the notion of an optimal transformation of one graph into another. The graph edit distance may be seen as a generalization of the string edit distance. In the case of strings, the set of editing operations consists of deletion, insertion and substitution of characters, and in the case of graphs, we consider deletion, insertion and substitution of vertices and edges. Below we cite the definition of the graph edit distance used in [Neuhaus and Bunke, 2007]

Definition 1 *Given two graphs G and H, the graph edit distance between G and H is defined by*

$$ged(G,H) = \min_{(e_1,\ldots,e_k)\in P(G,H)} \sum_{1}^{k} c(e_i)\,, \tag{1.7}$$

where $P(G,H)$ defines the set of all possible transformations $G \rightarrow H$, and $c(e_1)$ denotes the costs of editing operations: deletion, insertion and substitution of graph vertices and edges.

Each edit operation has an associated cost. Usually, these costs are defined as functions of edge and vertex labels. A natural hypothesis about edit operations is that a simple operation is always preferable to a sequence. For instance, substitution of vertex g by vertex h may be done via "substitution" or via "deletion+insertion", and a reasonable assumption is that the "substitution" cost should be smaller than the total "deletion+insertion" cost. Under this hypothesis, the notion of optimal edit path loses the idea of an ordered set of edit operations, all edit transformations may be applied simultaneously. Now it becomes clear how the graph edit distance may be related to formulation (1.5). Substitution of vertex g by vertex h means that these two vertices are matched to each other, the same is true for edges. When we insert a new vertex(edge) in graph G and match it to an existing vertex (edge) in graph H, it is somehow equivalent to deletion of vertices (edges) in graph H. Finally, the graph edit distance transformation may be seen as a matching of vertices (edges) which are chosen to be substituted, and then deletion of all unmatched elements in both graphs. Deletions in both graphs may be seen as matching of deleted vertices

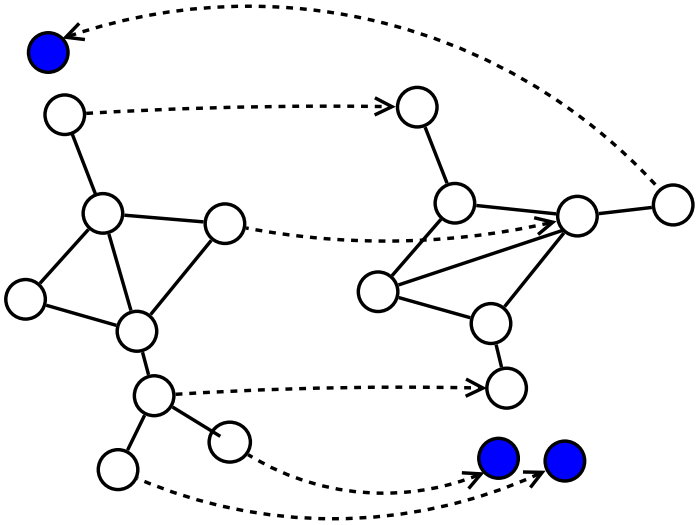


Figure 1.4: The graph edit path as simultaneous matching of all graph vertices and edges. Substitution operations correspond to matching of vertices and edges, vertex deletions correspond to alignment with dummy vertices (blue), edge deletion corresponds to alignment with nothing.

to dummy vertices. Figure 1.3.5 gives an illustration of this schema.

To finish the reformulation we have to clarify what happens with insertion, deletion and substitution costs when we reformulate (1.7) as (1.5). Usually, edit operation costs are defined as functions of vertex and edge labels. Vertex operation costs may be encoded directly in the matrix C (see (1.5)), the situation with edges is a bit more complicated. If edge labels are real valued weights, and substitution and deletion costs are defined as

$$\text{subst}(g_i, h_j) = (g_i - h_j)^2,\ \text{del}(g_i) = g_i^2,\ \text{del}(h_i) = h_i^2\,,$$

then the objective function defined in (1.5) equals the total cost of the edit path transformation. If, for instance, the costs of edit operations are defined as (see [Ambauen et al., 2003])

$$\text{subst}(g_i, h_j) = |g_i - h_j|,\ \text{del}(g_i) = |g_i|,\ \text{del}(h_i) = |h_i|\,,$$

then we have to replace the Frobenius norm in (1.5) by the l_1 norm (see Section 1.3.4). A similar adaptation of the objective function (1.5) should be made in the case of alternative edit operation costs.

1.3.6 Complexity of the graph matching problem

In some special cases, for example, when we restrict graph matching to graph isomorphism, it is difficult to say what the complexity of the corresponding problem is. For the moment, there is no known polynomial algorithm for graph isomorphism, and at the same time we do not know whether this problem is NP-hard or not. However in the general case, the graph matching problem is NP-hard, this fact follows naturally from the equivalency of the quadratic assignment problem and graph matching of weighted directed graphs.

1.4 Early history of graph matching

In this section we try to trace the evolution of the ideas related to the problem of graph matching. This objective is rather ambiguous and vague, but we believe it may be interesting to trace back how people came to the idea of graph comparison.

The commonly accepted view is that the first paper related to graph theory was the paper on the Seven Bridges of Königsberg written by Leonard Euler in 1735. At that time, the term "graph" had not been introduced, but the problem discussed in this paper is an example of a graph related problem. The term "graph" was introduced much later by James Sylvester in his article *"Chemistry and algebra", Nature, 1895.* Interestingly, the roots of graph theory are closely related to what we see now as an application area for graph-based methods.

As mentioned in the previous section, the current formulation of the graph matching problem was given by Tsai and Fu [1979]. But from a more general point of view, the graph matching problem represents an approach to graph comparison, a problem which was known long before [Tsai and Fu, 1979]. In the previous section, we have shown that the graph isomorphism problem, the subgraph isomorphism problem and

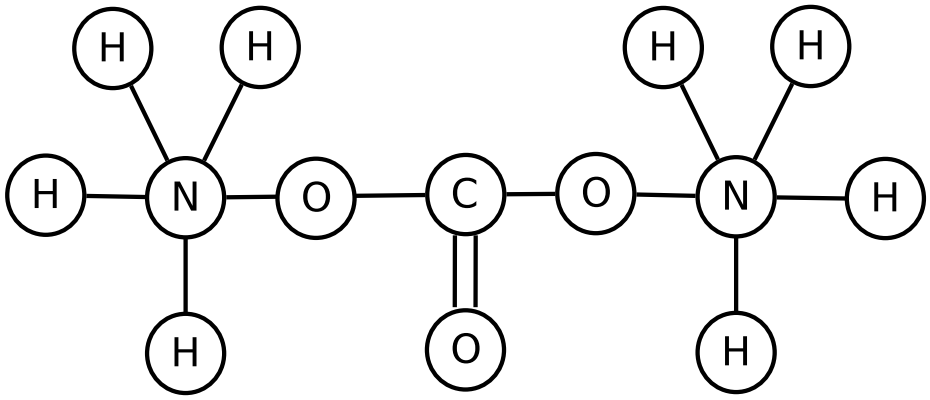
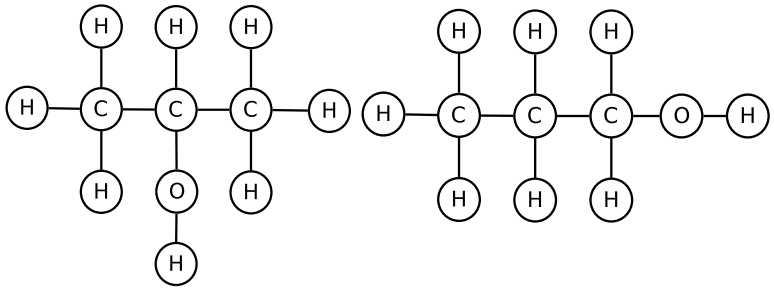


Figure 1.5: An example of Crum Brown's drawing: ammonic carbonate structure.

the problem of the maximum common subgraph may be seen as particular cases of the graph matching problem. These problems are related to different aspects of graph comparison such as whether two graphs have the same structure, whether one graph is a subgraph of another and so on. The graph isomorphism problem seems to be the most studied variant of all graph comparison formulations. The number of publications (sometimes containing wrong conclusions) on this topic is so big, that the graph isomorphism problem was called a graph isomorphism disease similarly to the four-colour problem [Read and Corneil, 1977].

It is hard to say when the graph isomorphism problem was formulated for the first time. By all appearances, the question about graph isomorphism arose at the same time that the term "graph" was introduced by J. Sylvester, and again it was closely related to existing problems in chemistry. In 1864 Alexander Crum Brown published a paper proposing a new system for the diagramic representation of molecules. His system is still popular and it corresponds exactly to how we draw graphs today (see Figure 1.5).

At this time several alternative representation systems were proposed by Couper, Loschmidt and Kekule, but the one by Crum Brown was the most successful. One of the reasons for this success is due to the fact that Crum Brown's system provides an explanation of the phenomena of molecular isomerism. Isomers are chemical compounds having the same chemical composition but different physico-chemical properties. For instance, Figure 1.6 shows two isomers: propan-1-ol and propan-2-ol known



(a) propylic alcohol (b) Friedel's alcohol

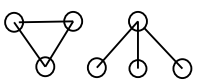
Figure 1.6: Examples of chemical isomers: propan-1-ol and propan-2-ol

at the time of Crum Brown as Friedel's alcohol and propylic alcohol.

Obviously, the question of chemical isomerism is equivalent to the graph isomorphism problem. Two substances with the same chemical composition represent different isomers if and only if their graphical notations correspond to non-isomorphic graphs.

The question of molecular isomers gave rise to graph enumeration theory. Roughly speaking, in the context of graph enumeration theory, we are interested in counting graphs with a particular property. The first works on this topic were done by Cayley [1874, 1859, 1875, 1877] where he proposed a method for counting different rooted trees with N vertices (Cayley used the term *knot*). So here, "different" means "non-isomorphic", Cayley applied his method to count the number of isomers C_nH_{2n+2}. Much later, Pòyla further developed this theory and proposed a basis for the general enumeration theory, a new branch in algebra.

About the same time that G.Polyà worked on his enumeration theory, Hassler Withney published a paper called *"Congruent graphs and connectivity of graphs", 1932*, where among other results he showed that two connected graphs are isomorphic if and only if their line graphs are isomorphic [1]. This paper is considered to be one of the first papers establishing a theoretical fact about graph isomorphism. Withney did not propose an algorithm for the graph isomorphism problem, but nevertheless his result is quite important for us, since it provides a better understanding of graph

[1] The only exception is K_3 and $K_{1,3}$ graphs 

isomorphism, and actually addresses the graph isomorphism problem in the form as it is known today.

The next important steps in the development of graph isomorphism algorithms are related to the evolution of computing hardware. In 1954 Gotusso and Santolini wrote the article *"A Fortran IV quasi-decision algorithm for the P-equivalence of two matrices"*. Two matrices G and H are P-equivalent if there exists a permutation matrix P such that

$$G = PHP^T.$$

While they did not consider the graph isomorphism problem directly, we know that the P-equivalence of graph adjacency matrices means that the corresponding graphs are isomorphic. Starting in the 1950s, a lot of papers were published proposing different algorithms for the graph isomorphism problem, we will not discuss them here, a good review of existing work on graph isomorphism was written by Read and Corneil [1977] and then completed by Gati [1979] and in a more recent technical report by Fortin [1996]. We would just like to mention two papers, one by Ray and Kirsch *"Finding Chemical Records by Digital Computers", 1957* which is considered to be one of the first papers in modern chemoinformatics, and another one by Sussenguth *"A graph theoretic algorithm for matching chemical structures", 1963* where he further developed the ideas of Ray and Kerisch and proposed to use (sub)-graph isomorphism methods for comparing chemical structures.

In the table below we cite key works which we believe had a significant impact on the understanding of the graph comparison problem. Further information on the historical development of graph theory and related concepts can be found in Biggs et al. [1976]; Diestel [2000]; Godsil and Royle [2000]; Bondy and Murty [1976].

Year	Authors	Paper & Comments
1735	L. Euler	The first publication on a graph related problem: *"Seven Bridges of Königsberg"*.
1864	A. Crum Brum	New graphic notation for chemical compounds (see Figure 1.5), understanding the phenomena of chemical isomerism.

Year	Authors	Paper & Comments
1875	J. Sylvester	His paper *"Chemistry and Algebra"* is about the similarity between the algebraic theory of binary quantics and the existing graphic representation of molecules. Introduction of the term *graph*.
1875	W. Clifford	His work was mainly concentrated on the invariants of binary quantics (*"Note on invariants of alternate numbers, used as a mean for determining the invariants and covariants of quantics in general"*, *"Binary forms of alternate variables"*), W. Clifford proposed the graph-based notation in this field.
1875-1877	A. Cayley	*"On the analytical forms called trees with application to the theory of chemical combinations", 1875*, *"On the number of univalent radicals C_nH_{2n-1}",1877*. Cayley was interested in how to count trees with a given number of vertices (knots) and the potential application of such methods to the enumeration of molecular isomers
1927	J. Redfield	His paper *"The theory of group-reduced distributions"* was one of the first papers in graph enumeration theory. While this paper is not well known, it forestalls to some extent a lot of ideas from G. Polyà's paper on graph enumeration.
1929	A. Lunn and J. Senior	In their paper *"Isomerism and Configurations"*, they discuss how the theory of permutation groups may be used to enumerate molecular isomers.
1930-1935	G. Polyà	In his paper *"A general combinatorial problem on groups of permutations and the calculation of the number of isomers of organic compounds"*, he builds the basis of modern graph enumeration theory.

Year	Authors	Paper & Comments
1930-1932	H. Whitney	Two papers of H. Whitney *"Non-separable and planar graphs", 1930* and *"Congruent graphs and the connectivity of graphs", 1932* address the problem of graph isomorphism as a pure mathematical problem and study different properties of isomorphic graphs.
1957	L. Ray, R. Kirsch	Their paper *"Finding chemical records by digital computers"* is considered as one of the first papers in chemoinformatics. They propose an information-retrieval process for the analysis of the dataset of known chemical compounds. While they paper is mainly conceptual, it includes some ideas about graph-based comparison of molecules.
1957	L. Gatusso, A. Santolini	Formally speaking, their work *"A Fortran IV quasi decision algorithm for the P-equivalence of two matrices"* is not a graph theoretical paper, but since graph isomorphism is equivalent to the P-equivalence of two matrices, this paper may be seen as one of the first papers providing an algorithm for the graph isomorphism problem.
1963	E. Sussenguth	E. Sussenguth further developed the ideas of L. Ray and R. Kirsch. His paper *"A graph theoretic algorithm for matching chemical structures"* is about how graph isomorphism algorithms may be used to detect similar chemical compounds.

Year	Authors	Paper & Comments
1950s-now		see Read and Corneil [1977]; Gati [1979] for the review of papers on the graph isomorphism problem. Papers on the application of graph comparison algorithms in chemistry may be found in [Willett, 2008]. Finally, a review of various graph matching algorithms proposed during the last thirty years was recently published by Conte et al. [2004].

1.4.1 Recent developments in graph matching

As mentioned in the previous section, a good review of existing graph matching methods may be found in Conte et al. [2004]. In addition to this paper, a review of graph isomorphism algorithms (and ideas related to graph comparison) published between 1950s and 1979 was proposed by Read and Corneil [1977] and further completed by Gati [1979].

We would like to mention here some papers to give a general idea about the main streams in the development of graph matching algorithms. Roughly speaking, there are three principal groups of algorithms for graph matching (they are all approximate since this problem is NP-hard in the general case). The first group consists of branch-and-bounds discrete optimization algorithms, examples are [Cordella et al., 1996, 2001; Tsai and Fu, 1979]. The second group represents various spectral approaches. The first spectral method for graph matching was proposed by Umeyama [1988], then similar approaches were developed by [Carcassoni and Hancock, 2003; Caelli and Kosinov, 2004; Xu and King, 1994]. A slightly different spectral approach is used by Leordeanu and Hebert [2005]; Cour et al. [2006]; Leordeanu et al. [2007].

The third group are methods based on different continuous relaxation of the original integer programming problem (1.4). Examples are [Gold and Rangarajan, 1996; Rangarajan and Mjolsness, 1996; Schellewald and Schnorr, 2005; Schellewald et al., 2001], in Chapter 2 we propose a new method based on a convex-concave continuous relaxation.

A more detailed description of the mentioned algorithms can be found in Sections 2.2.2 and 4.4

1.5 Applications of graph matching algorithms

In the previous sections, we have already seen that chemistry was a fertile ground for the development of graph theory and ideas related to graph comparison. In 1950s with the active use of computers in scientific computations, there was born a new branch of chemistry called chemoinformatics. And already at this time, some of the pioneering works in this area were dedicated to the application of different graph comparison algorithms, see, for example, [Sussenguth, 1963].

Another popular application area for graph comparison algorithms is computer vision including 2D and 3D analysis, image databank searches, video analysis, biometric identification and many other real-life problems. In computer vision, graphs are used as a universal tool for the representation of images of different kinds.

Recently, graph comparison algorithms have drawn a lot of interest in bioinformatics. Currently, with development of new technologies, we get more and more data on protein-protein interactions, gene co-expression and gene regulation. Such kinds of information may be naturally represented in the form of huge networks with nodes representing proteins or genes and edges representing their interactions. Comparison of such networks may be important for answering various questions about the functional roles of proteins, their interactions and to gain a better understanding of biological principles.

Other applications of graph matching algorithms include document processing where graph matching algorithms are used in OCR and electronic design automation verification software where graph matching algorithms are used for comparison of electronic circuits.

1.6 Graph matching, kernel methods and graph invariants: alternative approaches to graph comparison

An important class of graph comparison algorithms are kernels for graphs. This relatively new field studies the question of how to construct a positive-semidefinite similarity measure between graphs. The first paper proposing a kernel for graphs was written by Gärtner et al. [2002]. Random walk kernels were proposed by Gathner a year later. Marginalized kernels were generalized to graphs by Kashima et al. [2003] and further extended by Mahé et al. [2004].

The main difference between kernel methods and graph matching algorithms is that kernel methods do not produce an alignment between graphs. Roughly speaking, graph matching methods first search for an alignment between graphs and then measure the quality of the alignment produced, while kernel methods measure the similarity between graphs by "counting" the number of common small subgraphs. Therefore, kernels methods may be used in classification and clusterization tasks, but not in problems where we need to know which vertices (parts) of one graphs were matched to given vertices (parts) of another graph. At the same time, the graph matching distance (1.3) provides a true distance (or measure) on graphs, but this distance does not define directly a positive-definite similarity measure. Formally speaking, we can not use (1.3) in methods like SVM without additional tricks such as projection onto the set of positive-definite matrices.

We do not give a detailed description of kernel methods, a good review of existing works as well as a new unified framework for graph kernels may be found in [Vishwanathan et al., 2008]. Another interesting source of information is the recently published book of Neuhaus and Bunke [2007]. In this work, the authors discuss existing kernels for graphs and how new (or existing) kernels can be constructed on the basis of the graph edit distance.

Graph matching algorithms and kernel methods may be seen as two principal

groups of approaches to graph comparison. However, the most natural method consists in the construction of graph features which are invariant up to a permutation of graph vertices. This approach is often considered to be a particular case of the kernel-based approach since one usually uses the resulting feature vector as a part of a kernel-based algorithm. At the same time, in the context of graph invariant features, the main emphasis is on the construction of the invariants themselves, not on the development of sophisticated kernels. That is why we believe that graph invariants may be seen as a separate group of approaches. Simple examples of graph invariants are the number of graph vertices (edges), eigenvalues of the graph Laplacian matrix, graph diameter etc. More sophisticated features were proposed recently by Kondor and Borgwardt [2008]; Kondor et al. [2009] where the authors used elements of harmonic analysis on permutation groups to construct graph invariants.

Chapter 2

A path following algorithm for the graph matching problem

Preface

We propose a new graph matching algorithm based on convex-concave programming. The convex-concave programming formulation is obtained by rewriting the weighted graph matching problem as a least-square problem on the set of permutation matrices and relaxing it to two different optimization problems: a quadratic convex and a quadratic concave optimization problem on the set of doubly stochastic matrices. The concave relaxation has the same global minimum as the initial graph matching problem, but the search for its global minimum is also a hard combinatorial problem. We therefore construct an approximation of the concave problem solution by following a solution path of a convex-concave problem obtained by linear interpolation of the convex and concave formulations, starting from the convex relaxation. This method allows to easily integrate the information on graph label similarities into the optimization problem, and therefore to perform labeled weighted graph matching. The algorithm is compared with some of the best performing graph matching methods on four datasets: simulated graphs, QAPLib, retina vessel images and handwritten chinese characters. In all cases, the results are competitive with the state-of-the-art. This chapter is a slightly modified version of [Zaslavskiy et al., 2008c].

Our initial motivation was to create an efficient algorithm for graph comparison based on graph alignment. We started with ideas similar to the spectral approach of Umeyama [1988], but very soon realized that this approach had several major drawbacks: the graph spectral representation is not unique since eigenvectors are always defined up to a sign, and if there are close eigenvalues, then the corresponding eigenvectors are defined up to a rotation. So we decided to develop an alternative approach based on a continuous relaxation of the integer programming problem.

Originally, the new method was proposed for matching of **undirected** weighted labeled graphs. The symmetry of the graph adjacency matrix is crucial since the proposed concave relaxation can not be constructed for directed graphs with asymmetric adjacency matrices. Processing of directed graphs was an open question when the article was finished.

A possible way to run the proposed algorithm on directed graphs is to use the following transformation procedure which reduces the graph matching problem on $N \times N$ directed graphs to a graph matching problem on $2N \times 2N$ undirected graphs[1]. Let G and H denote two weighted[2] directed graphs to be matched. First, we split each vertex g_i in graph G into two vertices g_i^{in} and g_i^{out}, then we keep edges which are going into g_i as incident edges to vertex g_i^{in} (making them undirected) and edges which are coming out g_i as incident edges to g_i^{out} (and make them undirected). Finally we connect g_i^{in} and g_i^{our} by an edge with weight $M = 2(\sum_{i,j} G_{ij} + \sum_{i,j} H_{ij}) + \max_{i,j} G_{ij} + \max_{i,j} H_{ij}$. The new graph G' is an undirected graph with $2N$ vertices. The same transformation is performed on graph H. Now, the optimal graph matching between G' and H' under constraints that ^{in} vertices can be matched only to ^{in} vertices and ^{out} only to ^{out} is equivalent to the optimal matching of G and H.

It is easy to show that in the optimal alignment of G' and H' if g_i^{in} is matched to h_j^{in} then g_i^{out} is matched to h_j^{out} and vice versa (this means that M-edges are always aligned with each other). Indeed, if all M-edges are aligned only with M-edges then function $F_{G',H'}(P)$ may be upper bounded by $2(\sum_{ij} G_{ij} + \sum_{ij} H_{ij})$. At the same time

[1]Similar ideas are used in the traveling salesman problem to reduce asymmetric TSP to symmetric TSP, but we are not aware of any published transformation procedures for graph matching.

[2] We suppose that all edges have non-negative weights, otherwise we can add a positive constant to all edge weights to make them positive.

if at least one M-edge of graph G' is matched to an ordinary edge of H', then function $F_{G',H'}(P)$ may be lower bounded by $M - \max H_{ij} = 2(\sum_{i,j} G_{ij} + \sum_{i,j} H_{ij}) + \max_{i,j} G_{ij}$. Now, when $g_i^{in,out}$ are matched to $h_j^{in,out}$, it corresponds to matching $g_i \sim h_j$ in the original graphs and

$$2F_{G,H}^{opt} = F_{G',H'}^{opt}.$$

2.1 Introduction

During the last decades, many different algorithms for graph matching have been proposed. Because of the combinatorial nature of this problem, it is very hard to solve it exactly for large graphs, however some methods based on incomplete enumeration may be applied to search for an exact optimal solution in the case of small or sparse graphs. Some examples of such algorithms may be found in [Schmidt and Druffel, 1976; Ullmann, 1976; Cordella et al., 1999].

Another group of methods includes approximate algorithms which are supposed to be more scalable. The price to pay for the scalability is that the solution found is usually only an approximation of the optimal matching. Approximate methods may be divided into two groups of algorithms. The first group is composed of methods which use spectral representations of adjacency matrices, or equivalently embed the vertices into a Euclidean space where linear or nonlinear matching algorithms can be deployed. This approach was pioneered by Umeyama [1988], while further different methods based on spectral representations were proposed in [Shapiro and Brady, 1992; Carcassoni and Hancock, 2003; Luo and Hancock, 2000; Wang and Hancock, 2006; Caelli and Kosinov, 2004]. The second group of approximate algorithms is composed of algorithms which work directly with graph adjacency matrices, and typically involve a relaxation of the discrete optimization problem. The most effective algorithms were proposed in [Almohamad and Duffuaa, 1993; Gold and Rangarajan, 1996; Schellewald et al., 2001; Schellewald and Schnorr, 2005].

In this article we propose an approximate method for labeled weighted graph matching, based on a convex-concave programming approach which can be applied for matching of graphs of large sizes. Our method is based on a formulation of the

labeled weighted graph matching problem as a quadratic assignment problem (QAP) over the set of permutation matrices, where the quadratic term encodes the structural compatibility and the linear term encodes local compatibilities. We propose two relaxations of this problem, resulting in one quadratic convex and one quadratic concave minimization problem on the set of doubly stochastic matrices. While the concave relaxation has the same global minimum as the initial QAP, solving it is also a hard combinatorial problem. We find a local minimum of this problem by following a solution path of a family of convex-concave minimization problems, obtained by linearly interpolating between the convex and concave relaxations. Starting from the convex formulation with a unique local (and global) minimum, the solution path leads to a local optimum of the concave relaxation. We refer to this procedure as the PATH algorithm[3]. We perform an extensive comparison of this PATH algorithm with several state-of-the-art matching methods on small simulated graphs and various QAP benchmarks, and show that it consistently provides state-of-the-art performances while scaling to graphs of up to a few thousands vertices on a modern desktop computer. We further illustrate the use of the algorithm on two applications in image processing, namely the matching of retina images based on vessel organization, and the matching of handwritten chinese characters.

The rest of the chapter is organized as follows: Section 2.2 presents the mathematical formulation of the graph matching problem. In Section 2.3, we present our new approach. Then, in Section 2.4, we present the comparison of our method with Umeyama's algorithm [Umeyama, 1988] and the linear programming approach [Almohamad and Duffuaa, 1993] on the example of artificially simulated graphs. In Section 2.5, we test our algorithm on the QAP benchmark library, and we compare obtained results with the results published by Schellewald et al. [2001] for the QBP algorithm and graduated assignment algorithms. Finally, in Section 2.6 we present two examples of applications to real-world image processing tasks.

[3]The PATH algorithm as well as other referenced approximate methods are implemented in the software GraphM available at `http://cbio.ensmp.fr/graphm`

2.2 Problem description

A graph $G = (V, E)$ of size N is defined by a finite set of vertices $V = \{1, \ldots, N\}$ and a set of edges $E \subset V \times V$. We consider only undirected graphs with no self-loop, i.e., such that $(i, j) \in E \implies (j, i) \in E$ and $(i, i) \notin E$ for any vertices $i, j \in V$. Each such graph can be equivalently represented by a symmetric adjacency matrix A of size $|V| \times |V|$, where A_{Ia} is equal to one if there is an edge between vertex i and vertex j, and zero otherwise. An interesting generalization is a weighted graph which is defined by association of nonnegative real values w_{ij} (weights) to all edges of graph G. Such graphs are represented by real valued adjacency matrices A with $A_{ij} = w_{ij}$. This generalization is important because in many applications the graphs of interest have associated weights for all their edges, and taking into account these weights may be crucial in construction of efficient methods. In the following, when we talk about "adjacency matrix" we mean a real-valued "weighted" adjacency matrix.

Given two graphs G and H with the same number of vertices N, the problem of matching G and H consists in finding a correspondence between vertices of G and vertices of H which aligns G and H in some optimal way. We will consider in Section 1.3.3 an extension of the problem to graphs of different sizes. For graphs with the same size N, the correspondence between vertices is a permutation of N vertices, which can be defined by a permutation matrix P, i.e., a $\{0, 1\}$-valued $N \times N$ matrix with exactly one entry 1 in each column and each row. The matrix P entirely defines the mapping between vertices of G and vertices of H, P_{ij} being equal to 1 if the i-th vertex of G is matched to the j-th vertex of H, and 0 otherwise. After applying the permutation defined by P to the vertices of H we obtain a new graph isomorphic to H which we denote by $P(H)$. The adjacency matrix of the permuted graph, $A_{P(H)}$, is simply obtained from A_H by the equality $A_{P(H)} = PA_HP^T$.

In order to assess whether a permutation P defines a good matching between the vertices of G and those of H, a quality criterion must be defined. Although other choices are possible, we focus in this chapter on measuring the discrepancy between the graphs after matching, by the number of edges (in the case of weighted graphs, it will be the total weight of edges) which are present in one graph and not in the

other. In terms of adjacency matrices, this number can be computed as:

$$F_0(P) = ||A_G - A_{P(H)}||_F^2 = ||A_G - PA_HP^T||_F^2\,, \tag{2.1}$$

where $||.||_F$ is the Frobenius matrix norm defined by $\|A\|_F^2 = \mathrm{tr}A^TA = (\sum_i \sum_j A_{ij}^2)$. A popular alternative to the Frobenius norm formulation (2.1) is the 1-norm formulation obtained by replacing the Frobenius norm by the 1-norm $\|A\|_1 = \sum_i \sum_j |A_{ij}|$, which is equal to the square of the Frobenius norm $\|A\|_F^2$ when comparing $\{0,1\}$-valued matrices, but may differ in the case of general matrices.

Therefore, the problem of graph matching can be reformulated as the problem of minimizing $F_0(P)$ over the set of permutation matrices. This problem has a combinatorial nature and there is no known polynomial algorithm to solve it [Garey and Johnson, 1979]. It is therefore very hard to solve it in the case of large graphs, and numerous approximate methods have been developed.

2.2.1 Permutation matrices

Before describing how we propose to solve (2.1) and (1.5), we first introduce some notations and bring to notice some important characteristics of these optimization problems. They are defined on the set of permutation matrices, which we denoted by $\mathcal{P}$. The set $\mathcal{P}$ is a set of extreme points of the set of doubly stochastic matrices, that is, matrices with nonnegative entries and with row sums and column sums equal to one: $\mathcal{D} = \{A : A1_N = 1_N,\ A^T1_N = 1_N,\ A \geq 0\}$, where 1_N denotes the N-dimensional vector of all ones [Borwein and Lewis, 2000]. This shows that when a linear function is minimized over the set of doubly stochastic matrices $\mathcal{D}$, a solution can always be found in the set of permutation matrices. Consequently, minimizing a linear function over $\mathcal{P}$ is in fact equivalent to a linear program and can thus be solved in polynomial time by, e.g., interior point methods [Boyd and Vandenberghe, 2004]. In fact, one of the most efficient methods to solve this problem is the Hungarian algorithm, which uses a specific primal-dual strategy to solve this problem in $O(N^3)$ [McGinnis, 1983]. Note that the Hungarian algorithm allows to avoid the generic $O(N^7)$ complexity associated with a linear program with N^2 variables.

At the same time $\mathcal{P}$ may be considered as a subset of orthonormal matrices $\mathcal{O} = \{A : A^T A = I\}$ of $\mathcal{D}$ and in fact $\mathcal{P} = \mathcal{D} \cap \mathcal{O}$. An (idealized) illustration of these sets is presented in Figure 2.1: the discrete set $\mathcal{P}$ of permutation matrices is the intersection of the convex set $\mathcal{D}$ of doubly stochastic matrices and the manifold $\mathcal{O}$ of orthogonal matrices.

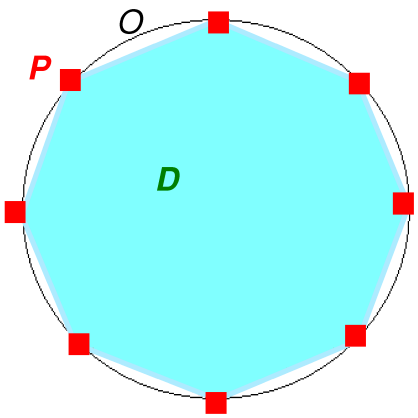


Figure 2.1: Relation between three matrix sets. $\mathcal{O}$—set of orthogonal matrices, $\mathcal{D}$ — set of doubly stochastic matrices, $\mathcal{P} = D \cap O$—set of permutation matrices.

2.2.2 Approximate methods: existing works

A good review of graph matching algorithms may be found in [Conte et al., 2004]. Here, we only present a brief description of some approximate methods which illustrate well ideas behind two subgroups of these algorithms. As mentioned in the introduction, one popular approach to find approximate solutions to the graph matching problem is based on the spectral decomposition of the adjacency matrices of the graphs to be matched. In this approach, the singular value decompositions of the graph adjacency matrices are used:

$$A_G = U_G \Lambda_G U_G^T, \;\; A_H = U_H \Lambda_H U_H^T,$$

where the columns of the orthogonal matrices U_G and U_H consist of eigenvectors of A_G and A_H respectively, and Λ_G and Λ_H are diagonal matrices of eigenvalues.

If we consider the rows of eigenvector matrices U_G and U_H as graph node coordinates in eigenspaces, then we can match the vertices with similar coordinates through a variety of methods [Umeyama, 1988; Carcassoni and Hancock, 2003; Caelli and Kosinov, 2004]. However, these methods suffer from the fact that the spectral embedding of graph vertices is not uniquely defined. First, the unit norm eigenvectors are at most defined up to a sign flip and we have to choose their signs synchronously. Although it is possible to use some normalization convention, such as choosing the sign of each eigenvector in such a way that the biggest component is always positive, this usually does not guarantee a perfect sign synchronization, in particular in the presence of noise. Second, if the adjacency matrix has multiple eigenvalues, then the choice of eigenvectors becomes arbitrary within the corresponding eigen-subspace, as they are defined only up to rotations [Golub and Loan, 1996].

One of the first spectral approximate algorithms was presented by Umeyama [1988]. To avoid the ambiguity of eigenvector selection, Umeyama proposed to consider the absolute values of eigenvectors. According to this approach, the correspondence between graph nodes is established by matching the rows of $|U_G|$ and $|U_H|$ (which are defined as matrices of absolute values). The criterion of optimal matching is the total distance between matched rows, leading to the optimization problem:

$$\min_{P \in \mathcal{P}} \; \| \; |U_G| - P|U_H| \; \|_F \, ,$$

or equivalently:

$$\max_{P \in \mathcal{P}} \; \mathrm{tr}(|U_H||U_G|^T P) \, . \tag{2.2}$$

The optimization problem (2.2) is a linear program on the set of permutation matrices which can be solved by the Hungarian algorithm in $O(N^3)$ [McGinnis, 1983; Kuhn, 1955].

Recently Leordeanu and Hebert [2005]; Cour et al. [2006]; Leordeanu et al. [2007] proposed an alternative spectral approach. They reformulated the graph matching

problem as a quadratic assignment problem

$$\begin{aligned} &\max_x x^T M x \\ &\text{subject to } Sx = 1,\ x \in \{0,1\}^{N^2}, \end{aligned} \tag{2.3}$$

where M is the square matrix representing pairwise similarties between graph edges $M_{(i,j),(i',j')} = sim(G_{ij}, H_{ij})$ and x encodes a consistent assignment between graph edges $x_{i+jN} = 1$ if vertex i of graph G is matched to vertex j of graph H (x corresponds to vec(P) and S controls that x is consistent). If sim is the simple scalar product between edge weights i.e. $M = A_G \otimes A_H$, then (2.3) is eqiuvalent to the classical formulation of graph matching. Leordeanu and Hebert [2005] proposed a relaxation of (2.3), where optimization constraints are replaced by $||x||_2 = 1, x \in [0,1]^{N^2}$. The optimal solution of the resulting continious optimization problem is the first eigenvector of M (Raleigh's ratio theorem and Perron-Frobenius theorem). Finally, to construct the optimal assignment we can project x on the set of permutation matrices. Again if sim is the simple scalar product, then x_{opt} can be expressed as the tensor product of the first eigenvector of A_H and the first eigenvector of A_G, otherwise, in the general case, we can compute the first eigenvector of M in $O((N^2)^{3/2})$.

The second group of approximate methods consists of algorithms which work directly with the objective function in (2.1), and typically involve various relaxations to optimizations problems that can be efficiently solved. An example of such an approach is the linear programming method proposed by Almohamad and Duffuaa [1993]. They considered the 1-norm as the matching criterion for a permutation matrix $P \in \mathcal{P}$:

$$F_0'(P) = ||A_G - P A_H P^T||_1 = ||A_G P - P A_H||_1.$$

The linear program relaxation is obtained by optimizing $F_0'(P)$ on the set of doubly stochastic matrices $\mathcal{D}$ instead of $\mathcal{P}$:

$$\min_{P \in \mathcal{D}} F_0'(P), \tag{2.4}$$

where the 1-norm of a matrix is defined as the sum of the absolute values of all the elements of a matrix. A priori the solution of (2.4) is an arbitrary doubly stochastic matrix $X \in \mathcal{D}$, so the final step is a projection of X on the set of permutation matrices (we let denote $\Pi_{\mathcal{P}} X$ the projection of X onto $\mathcal{P}$) :

$$P^* = \Pi_{\mathcal{P}} X = \arg\min_{P \in \mathcal{P}} \; ||P - X||_F^2 \,,$$

or equivalently:

$$P^* = \arg\max_{P \in \mathcal{P}} X^T P \,. \tag{2.5}$$

The projection (2.5) can be performed with the Hungarian algorithm, with a complexity cubic in the dimension of the problem. The main disadvantage of this method is that the dimensionality (i.e., number of variables and number of constraints) of the linear program (2.5) is $O(N^2)$, and therefore it is quite hard to process graphs of size more than one hundred nodes.

Other convex relaxations of (2.1) can be found in [Schellewald et al., 2001] and [Gold and Rangarajan, 1996]. In the next section we describe our new algorithm which is based on the technique of convex-concave relaxations of the initial problems (2.1) and (1.5).

2.3 Convex-concave relaxation

Let us start the description of our algorithm for unlabeled weighted graphs. The generalization to labeled weighted graphs is presented in Section 2.3.7. The graph matching criterion we consider for unlabeled graphs is the square of the Frobenius norm of the difference between adjacency matrices (2.1). Since permutation matrices are also orthogonal matrices (i.e., $PP^T = I$ and $P^T P = I$), we can rewrite $F_0(P)$ on $\mathcal{P}$ as follows:

$$\begin{aligned} F_0(P) &= \|A_G - P A_H P^T\|_F^2 = \|(A_G - P A_H P^T) P\|_F^2 \\ &= \|A_G P - P A_H\|_F^2 \,. \end{aligned}$$

The graph matching problem is then the problem of minimizing $F_0(P)$ over $\mathcal{P}$, which we call **GM**:

$$\textbf{GM}: \min_{P \in \mathcal{P}} \; F_0(P)\,. \tag{2.6}$$

2.3.1 Convex relaxation

A first relaxation of **GM** is obtained by expanding the convex quadratic function $F_0(P)$ on the set of doubly stochastic matrices $\mathcal{D}$:

$$\textbf{QCV}: \min_{P \in \mathcal{D}} \; F_0(P)\,. \tag{2.7}$$

The **QCV** problem is a convex quadratic program that can be solved in polynomial time, e.g., by the Frank-Wolfe algorithm [Frank and Wolfe, 1956] (see Section 2.3.5 for more details). However, the optimal value is usually not an extreme points of $\mathcal{D}$, and therefore not a permutation matrix. If we want to use only **QCV** for the graph matching problem, we therefore have to project its solution on the set of permutation matrices, and to make, e.g., the following approximation:

$$\arg\min_{\mathcal{P}} \; F_0(P) \approx \Pi_{\mathcal{P}} \arg\min_{\mathcal{D}} \; F_0(P)\,. \tag{2.8}$$

Although the projection $\Pi_{\mathcal{P}}$ can be made efficiently in $O(N^3)$ by the Hungarian algorithm, the difficulty with this approach is that if $\arg\min_{\mathcal{D}} \; F_0(P)$ is far from $\mathcal{P}$ then the quality of the approximation (2.8) may be poor: in other words, the work performed to optimize $F_0(P)$ is partly lost by the projection step which is independent of the cost function. The PATH algorithm that we present later can be thought of as a improved projection step that takes into account the cost function in the projection.

2.3.2 Concave relaxation

We now present a second relaxation of **GM**, which results in a concave minimization problem. For that purpose, let us introduce the diagonal degree matrix D of an adjacency matrix A, which is the diagonal matrix with entries $D_{ii} = d(i) = \sum_{i=1}^{N} A_{ij}$ for $i = 1, \ldots, N$, as well as the Laplacian matrix $L = D - A$. A having only nonnegative

entries, it is well-known that the Laplacian matrix is positive semidefinite [Chung, 1997]. We can now rewrite $F_0(P)$ as follows:

$$\begin{aligned} F_0(P) =& ||A_G P - P A_H||_F^2 \\ =& ||(D_G P - P D_H) - (L_G P - P L_H)||_F^2 \\ =& ||D_G P - P D_H||_F^2 \\ & - 2\mathrm{tr}[(D_G P - P D_H)^T (L_G P - P L_H)] \\ & + ||L_G P - P L_H||_F^2 . \end{aligned} \tag{2.9}$$

Let us now consider more precisely the second term in this last expression:

$$\begin{aligned} & \mathrm{tr}[(D_G P - P D_H)^T (L_G P - P L_H)] \\ & = \underbrace{\mathrm{tr} P P^T D_G^T L_G}_{\sum d_G^2(i)} + \underbrace{\mathrm{tr} L_H D_H^T P^T P}_{\sum d_H^2(i)} - \underbrace{\mathrm{tr} P^T D_G^T P L_H}_{\sum d_G(i) d_{P(H)}(i)} \\ & \qquad\qquad - \underbrace{\mathrm{tr} D_H^T P^T L_G P}_{\sum d_{P(H)}(i) d_G(i)} \\ & = \sum (d_G(i) - d_{P(H)}(i))^2 = \|D_G - D_{P(H)}\|_F^2 \\ & = \|D_G P - P D_H\|_F^2 . \end{aligned} \tag{2.10}$$

Plugging (2.10) into (2.9) we obtain

$$\begin{aligned} F_0(P) &= \|D_G P - P D_H\|_F^2 - 2\|D_G P - P D_H\|_F^2 \\ &\quad + \|L_G P - P L_H\|_F^2 \\ &= -\|D_G P - P D_H\|_F^2 + \|L_G P - P L_H\|_F^2 \\ &= -\sum_{i,j} P_{ij} (D_G(j) - D_H(i))^2 + \mathrm{tr}(\underbrace{P P^T}_{I} L_G^T L_G) \\ &\qquad + \mathrm{tr}(L_H^T \underbrace{P^T P}_{I} L_H) - 2 \underbrace{\mathrm{tr}(P^T L_G^T P L_H)}_{\mathrm{vec}(P)^T (L_H^T \otimes L_G^T) \mathrm{vec}(P)} \\ &= -\mathrm{tr}(\Delta P) + \mathrm{tr}(L_G^2) + \mathrm{tr}(L_H^2) \\ &\qquad\qquad - 2\mathrm{vec}(P)^T (L_H^T \otimes L_G^T) \mathrm{vec}(P) , \end{aligned} \tag{2.11}$$

where we introduced the matrix $\Delta_{i,j} = (D_H(j,j) - D_G(i,i))^2$ and we used $\otimes$ to denote the Kronecker product of two matrices (definition of the Kronecker product and some important properties may be found in the appendix 2.B).

Let us denote $F_1(P)$ the part of (2.11) which depends on P:

$$F_1(P) = -\mathrm{tr}(\Delta P) - 2\mathrm{vec}(P)^T(L_H^T \otimes L_G^T)\mathrm{vec}(P).$$

From (2.11) we see that the permutation matrix which minimizes F_1 over $\mathcal{P}$ is the solution of the graph matching problem. Now, minimizing $F_1(P)$ over $\mathcal{D}$ gives us a relaxation of (2.6) on the set of doubly stochastic matrices. Since graph Laplacian matrices are positive semi-definite, the matrix $L_H \otimes L_G$ is also positive semidefinite as a Kronecker product of two symmetric positive semi-definite matrices [Golub and Loan, 1996]. Therefore the function $F_1(P)$ is concave on $\mathcal{D}$, and we obtain a concave relaxation of the graph matching problem:

$$\textbf{QCC}: \min_{P \in \mathcal{D}} F_1(P). \tag{2.12}$$

Interestingly, the global minimum of a concave function is necessarily located at a boundary of the convex set where it is minimized [Rockafeller, 1970], so the minimium of $F_1(P)$ on $\mathcal{D}$ is in fact in $\mathcal{P}$.

At this point, we have obtained two relaxations of **GM** as nonlinear optimization problems on $\mathcal{D}$: the first one is the convex minimization problem **QCV** (2.7), which can be solved efficiently but leads to a solution in $\mathcal{D}$ that must then be projected onto $\mathcal{P}$, and the other is the concave minimization problem **QCC** (2.12) which does not have an efficient (polynomial) optimization algorithm but has the same solution as the initial problem **GM**. We note that these convex and concave relaxation of the graph matching problem can be more generally derived for any quadratic assignment problems [Anstreicher and Brixius, 2001].

2.3.3 PATH algorithm

We propose to approximately solve **QCC** by tracking a path of local minima over $\mathcal{D}$ of a series of functions that linearly interpolate between $F_0(P)$ and $F_1(P)$, namely:

$$F_\lambda(P) = (1-\lambda)F_0(P) + \lambda F_1(P)\,,$$

for $0 \leq \lambda \leq 1$. For all $\lambda \in [0,1]$, F_λ is a quadratic function (which is in general neither convex nor concave for λ away from zero or one). We recover the convex function F_0 for $\lambda = 0$, and the concave function F_1 for $\lambda = 1$. Our method searches sequentially local minima of F_λ, where λ moves from 0 to 1. More precisely, we start at $\lambda = 0$, and find the unique local minimum of F_0 (which is in this case its unique global minimum) by any classical QP solver. Then, iteratively, we find a local minimum of $F_{\lambda+d\lambda}$ given a local minimum of F_λ by performing a local optimization of $F_{\lambda+d\lambda}$ starting from the local minimum of F_λ, using for example the Frank-Wolfe algorithm. Repeating this iterative process for $d\lambda$ small enough we obtain a path of solutions $P^*(\lambda)$, where $P^*(0) = \arg\min_{P\in\mathcal{D}} F_0(P)$ and $P^*(\lambda)$ is a local minimum of F_λ for all $\lambda \in [0,1]$. Noting that any local minimum of the concave function F_1 on $\mathcal{D}$ is in $\mathcal{P}$, we finally output $P^*(1) \in \mathcal{P}$ as an approximate solution of **GM**.

Our approach is similar to graduated non-convexity [Blake and Zisserman, 1987]: this approach is often used to approximate the global minimum of a non-convex objective function. This function consists of two part, the convex component, and non-convex component, and the graduated non-convexity framework proposes to track the linear combination of the convex and non-convex parts (from the convex relaxation to the true objective function) to approximate the minimum of the non-convex function. The PATH algorithm may indeed be considered as an example of such an approach. However, the main difference is the construction of the objective function. Unlike Blake and Zisserman [1987], we construct two relaxations of the initial optimization problem, which lead to the same value on the set of interest ($\mathcal{P}$), the goal being to choose convex/concave relaxations which approximate in the best way the objective function on the set of permutation matrices.

The pseudo-code for the PATH algorithm is presented in Figure 2.2. The rationale behind it is that among the local minima of $F_1(P)$ on $\mathcal{D}$, we expect the one connected to the global minimum of F_0 through a path of local minima to be a good approximation of the global minima. Such a situation is for example shown in Figure 2.3, where in 1 dimension the global minimum of a concave quadratic function on an interval (among two candidate points) can be found by following the path of local minima connected to the unique global minimum of a convex function.

1. **Initialization**:
   (a) $\lambda := 0$
   (b) $P^*(0) = \arg\min F_0$ — convex optimization problem, global minimum is found by Frank-Wolfe algorithm.
2. **Cycle over** λ:
   while $\lambda < 1$
   (a) $\lambda_{new} := \lambda + d\lambda$
   (b) if $|F_{\lambda_{new}}(P^*(\lambda)) - F_\lambda(P^*(\lambda))| < \epsilon_\lambda$ then
   $\lambda = \lambda_{new}$
   else $P^*(\lambda_{new}) = \arg\min F_{\lambda_{new}}$ is found
   by Frank-Wolfe starting from $P^*(\lambda)$
   $\lambda = \lambda_{new}$
3. **Output**: $P^{out} := P^*(1)$

Figure 2.2: Schema of the PATH algorithm

More precisely, and although we do not have any formal result about the optimality of the PATH optimization method (beyond the lack of global optimality, see Appendix 2.A), we can mention a few interesting properties of this method:

- We know from (2.11) that for $P \in \mathcal{P}$, $F_1(P) = F_0(P) - \kappa$, where $\kappa = \mathrm{tr}(L_G^2) + \mathrm{tr}(L_H^2)$ is a constant independent of P. As a result, it holds for all $\lambda \in [0, 1]$ that, for $P \in \mathcal{P}$:

$$F_\lambda(P) = F_0(P) - \lambda\kappa\,.$$

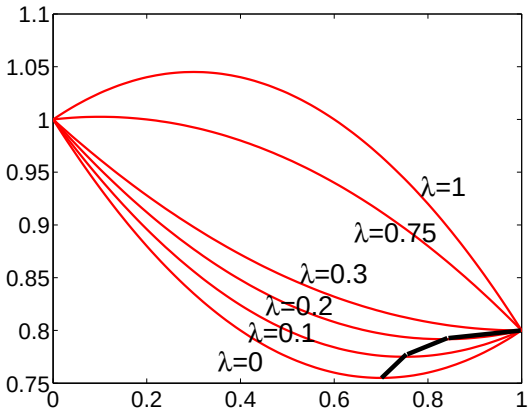


Figure 2.3: Illustration for path optimization approach. F_0 ($\lambda = 0$) — initial convex function, F_1 ($\lambda = 1$) — initial concave function, bold black line — path of function minima $P^*(\lambda)$ ($\lambda = 0 \ldots 0.1 \ldots 0.2 \ldots 0.3 \ldots 0.75 \ldots 1$)

This shows that if for some λ the global minimum of $F_\lambda(P)$ over $\mathcal{D}$ lies in $\mathcal{P}$, then this minimum is also the global minimum of $F_0(P)$ over $\mathcal{P}$ and therefore the optimal solution of the initial problem. Hence, if for example the global minimum of F_λ is found on $\mathcal{P}$ by the PATH algorithm (for instance, if F_λ is still convex), then the PATH algorithm leads to the global optimum of F_1. This situation can be seen in the toy example in Figure 2.3 where, for $\lambda = 0.3$, F_λ has its unique minimum at the boundary of the domain.

- The sub-optimality of the PATH algorithm comes from the fact that, when λ increases, the number of local minima of F_λ may increase and the sequence of local minima tracked by PATH may not be global minima. However we can expect the local minima followed by the PATH algorithm to be interesting approximations for the following reason. First observe that if P_1 and P_2 are two local minima of F_λ for some $\lambda \in [0, 1]$, then the restriction of F_λ to (P_1, P_2) being a quadratic function it has to be concave and P_1 and P_2 must be on the boundary of $\mathcal{D}$. Now, let λ_1 be the smallest λ such that F_λ has several local minima on $\mathcal{D}$. If P_1 denotes the local minima followed by the PATH algorithm, and P_2 denotes the "new" local minimum of F_{λ_1}, then necessarily

the restriction of F_{λ_1} to (P_1, P_2) must be concave and have a vanishing derivative in P_2 (otherwise, by continuity of F_λ in λ, there would be a local minimum of F_λ near P_2 for λ slightly smaller than λ_1). Consequently we necessarily have $F_{\lambda_1}(P_1) < F_{\lambda_1}(P_2)$. This situation is illustrated in Figure 2.3 where, when the second local minimum appears for $\lambda = 0.75$, it is worse than the one tracked by the PATH algorithm. More generally, when "new" local minima appear, they are strictly worse than the one tracked by the PATH algorithm. Of course, they may become better than the PATH solution when λ continues to increase.

Of course, in spite of these justifications, the PATH algorithm only gives an approximation of the global minimum in the general case. In Appendix 2.A, we provide two simple examples when the PATH algorithm respectively succeeds and fails to find the global minimum of the graph matching problem.

2.3.4 Numerical continuation method interpretation

Our path following algorithm may be considered as a particular case of numerical continuation methods (sometimes called path following methods) [Allgower and K.Georg, 1990]. These allow to estimate curves given in the following implicit form:

$$T(u) = 0 \text{ where } T \text{ is a mapping } T : R^{K+1} \to R^K \,. \tag{2.13}$$

In fact, our PATH algorithm corresponds to a particular implementation of the so-called Generic Predictor Corrector Approach [Allgower and K.Georg, 1990] widely used in numerical continuation methods.

In our case, we have a set of problems $\min_{P \in \mathcal{D}} (1-\lambda)F_0(P) + \lambda F_1(P)$ parametrized by $\lambda \in [0,1]$. In other words for each λ we have to solve the following system of Karush-Kuhn-Tucker (KKT) equations:

$$\begin{aligned} (1-\lambda)\nabla_P F_0(P) + \lambda \nabla_P F_1(P) + B^T\nu + \mu_S &= 0\,, \\ B\mathrm{P} - 1_{2N} &= 0\,, \\ P_S &= 0\,, \end{aligned}$$

where S is a set of active constraints, i.e., of pairs of indices (i, j) that satisfy $P_{ij} = 0$, $B\mathrm{P} - 1_{2N} = 0$ codes the conditions $\sum_j P_{ij} = 1\ \forall i$ and $\sum_i P_{ij} = 1\ \forall j$, ν and μ_S are dual variables. We have to solve this system for all possible sets of active constraints S on the open set of matrices P that satisfy $P_{i,j} > 0$ for $(i, j) \notin S$, in order to define the set of stationary points of the functions F_λ. Now if we let $T(P, \nu, \mu, \lambda)$ denote the left-hand part of the KKT equation system then we have exactly (2.13) with $K = N^2 + 2N + \#S$. From the implicit function theorem [Milnor, 1969], we know that for each set of constraints S,

$$\begin{aligned} W_S = &\{(P, \nu, \mu_S, \lambda) : T(P, \nu, \mu_S, \lambda) = 0 \text{ and} \\ &T'(P, \nu, \mu_S, \lambda) \text{ has the maximal possible rank}\} \end{aligned}$$

is a smooth 1-dimensional curve or the empty set and can be parametrized by λ. In term of the objective function $F_\lambda(P)$, the condition on $T'(P, \nu, \mu_S, \lambda)$ may be interpreted as a prohibition for the projection of $F_\lambda(P)$ on any feasible direction to be a constant. Therefore the whole set of stationary points of $F_\lambda(P)$ when λ is varying from 0 to 1 may be represented as a union $W(\lambda) = \cup_S W_S(\lambda)$ where each $W_S(\lambda)$ is homotopic to a 1-dimensional segment. The set $W(\lambda)$ may have quite complicated form. Some of $W_S(\lambda)$ may intersect each other, in this case we observe a bifurcation point, some of $W_S(\lambda)$ may connect each other, in this case we have a transformation point of one path into another, some of $W_S(\lambda)$ may appear only for $\lambda > 0$ and/or disappear before λ reaches 1. At the beginning the PATH algorithm starts from $W_\emptyset(0)$, then it follows $W_\emptyset(\lambda)$ until the border of $\mathcal{D}$ (or a bifurcation point). If such an event occurs before $\lambda = 1$ then PATH moves to another segment of solutions corresponding to different constraints S, and keeps moving along segments and sometimes jumping between segments until $\lambda = 1$. As we said in the previous section one of the interesting properties of PATH algorithm is the fact that if $W^*_S(\lambda)$ appears only when $\lambda = \lambda_1$ and $W^*_S(\lambda_1)$ is a local minimum then the value of the objective function F_{λ_1} in $W^*_S(\lambda_1)$ is greater than in the point traced by the PATH algorithm.

2.3.5 Some implementation details

In this section we provide a few details relevant for the efficient implementation of the PATH algorithms.

Frank-Wolfe Among the different optimization techniques for the optimization of $F_\lambda(P)$ starting from the current local minimum tracked by the PATH algorithm, we use in our experiments the Frank-Wolfe algorithm which is particularly suited to optimization over doubly stochastic matrices [Bertsekas, 1999]. The idea of the this algorithm is to sequentially minimize linear approximations of $F_0(P)$. Each step includes three operations:

1. estimation of the gradient $\nabla F_\lambda(P_n)$,
2. resolution of the linear program $P_n^* = \arg\min_{P\in D}\langle \nabla F_\lambda(P_n), P\rangle$,
3. line search: finding the minimum of $F_\lambda(P)$ on the segment $[P_n\ P_n^*]$.

An important property of this method is that the second operation can be done efficiently by the Hungarian algorithm, in $O(N^3)$.

Efficient gradient computations Another essential point is that we do not need to store matrices of size $N^2 \times N^2$ for the computation of $\nabla F_1(P)$, because the tensor product in $\nabla F_1(P) = -\text{vec}(\Delta^T) - 2(L_H^T \otimes L_G^T)\text{vec}(P)$ can be expressed in terms of $N \times N$ matrix multiplication:

$$\begin{aligned}\nabla F_1(P) &= -\text{vec}(\Delta^T) - 2(L_H^T \otimes L_G^T)\text{vec}(P)\\ &= -\text{vec}(\Delta^T) - 2\text{vec}(L_G^T P L_H).\end{aligned}$$

The same thing may be done for the gradient of the convex component

$$\begin{aligned}&\nabla F_0(P) = \nabla[\text{vec}(P)^T Q \text{vec}(P)]\\ &\quad \text{where } Q = (I \otimes A_G - A_H^T \otimes I)^T (I \otimes A_G - A_H^T \otimes I)\\ &\nabla F_0(P) = 2Q\text{vec}(P)\\ &\qquad = 2\text{vec}(A_G^2 P - A_G^T P A_H^T - A_G P A_H + P A_H^2)\end{aligned}$$

Initialization The proposed algorithm can be accelerated by the application of Newton algorithm as the first step of **QCV** (minimization of $F_0(P)$). First, let us rewrite the **QCV** problem as follows:

$$\min_{P\in\mathcal{D}} \|A_G P - P A_H\|_F^2 \Leftrightarrow$$

$$\min_{P\in\mathcal{D}} \text{vec}(P)^T Q \text{vec}(P) \Leftrightarrow \begin{cases} \min_P \text{vec}(P)^T Q \text{vec}(P) \\ \text{such that} \\ B\text{vec}(P) = 1_{2N} \\ \text{vec}(P) \geq 0_{N^2} \end{cases} \tag{2.14}$$

where B is the matrix which codes the conditions $\sum_j P_{i,j} = 1$ and $\sum_i P_{i,j} = 1$. The Lagrangian has the following form

$$\begin{aligned}\mathcal{L}(P,\nu,\lambda) =& \text{vec}(P)^T Q \text{vec}(P) + \nu^T (B\text{vec}(P) \\ & - 1_{2N}) + \mu^T \text{vec}(P),\end{aligned}$$

where ν and μ are Lagrange multipliers. Now we would like to use Newton method for constrained optimization [Bertsekas, 1999] to solve (2.14). Let μ_a denote the set of variables associated to the set of active constraints $\text{vec}(P) = 0$ at the current points, then the Newton step consist in solving the following system of equations:

$$\begin{bmatrix} 2Q & B^T & I_a \\ B & 0 & 0 \\ I_a & 0 & 0 \end{bmatrix} \begin{bmatrix} \text{vec}(P) \\ \nu \\ \mu_a \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} \begin{matrix} N^2 \text{ elements,} \\ 2N \text{ elements,} \\ \#\text{ of act. ineq. cons.} \end{matrix} \tag{2.15}$$

More precisely we have to solve (2.15) for P. The problem is that in general situations this problem is computationally demanding because it involves the inversion of matrices of size $O(N^2) \times O(N^2)$. In some particular cases, however, the Newton step becomes feasible. Typically, if none of the constraints $\text{vec}(P) \geq 0$ are active, then

(2.15) takes the following form[4]:

$$\begin{bmatrix} 2Q & B^T \\ B & 0 \end{bmatrix} \begin{bmatrix} \mathrm{vec}(P) \\ \nu \end{bmatrix} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \quad \begin{matrix} N^2 \text{ elements}, \\ 2N \text{ elements}. \end{matrix} \tag{2.16}$$

The solution is then obtained as follows:

$$\mathrm{vec}(P)_{KKT} = \frac{1}{2} Q^{-1} B^T (B Q^{-1} B^T)^{-1} 1_{2N}. \tag{2.17}$$

Because of the particular form of matrices Q and B, the expression (2.17) may be computed very simply with the help of Kronecker product properties in $O(N^3)$ instead of $O(N^6)$. More precisely, the first step is the calculation of $M = BQ^{-1}B^T$ where $Q = (I \otimes A_G - A_H^T \otimes I)^2$. The matrix Q^{-1} may be represented as follows:

$$Q^{-1} = (U_H \otimes U_G)(I \otimes \Lambda_G - \Lambda_H \otimes I)^{-2}(U_H \otimes U_G)^T. \tag{2.18}$$

Therefore the (i,j)-th element of M is the following product:

$$\begin{aligned} B_i Q^{-1} B_j^T = \mathrm{vec}(U_H^T \widetilde{B_i}^T U_G)^T)(\Lambda_G - \Lambda_H)^{-2} \\ \times \mathrm{vec}(U_G^T \widetilde{B_j}^T U_H), \end{aligned} \tag{2.19}$$

where B_i is the i-th row of B and $\widetilde{B_i}$ is B_i reshaped into a $N \times N$ matrix. The second step is an inversion of the $2N \times 2N$ matrix M, and a sum over columns $M^s = M^{-1} 1_{2N}$. The last step is a multiplication of Q^{-1} by $B^T M^s$, which can be done with the same tricks as the first step. The result is the value of matrix P_{KKT}. We then have two possible scenarios:

1. If $P_{KKT} \in \mathcal{D}$, then we have found the solution of (2.14).

2. Otherwise we take the point of intersection of the line (P_0, P_{KKT}) and the border $\partial\mathcal{D}$ as the next point and we continue with Frank-Wolfe algorithm. Unfortunately we can do the Newton step only once, then some of $P \geq 0$ constraints

[4]It is true if we start our algorithm, for example, from the constant matrix $P_0 = \frac{1}{N} 1_N 1_N^T$.

become active and efficient calculations are not feasible anymore. But even in this case the Newton step is generally very useful because it decreases a lot the value of the objective function.

$d\lambda$-adaptation strategy In practice, we found it useful to have the parameter $d\lambda$ in the algorithm of Figure 2.2 vary between iterations. Intuitively, $d\lambda$ should depend on the form of the objective function as follows: if $F_\alpha^\lambda(P)$ is smooth and if increasing the parameter λ does not change a lot the form of the function, then we can afford large steps, in contrast, we should do a lot of small steps in the situation where the objective function is very sensitive to changes in the parameter λ. The adaptive scheme we propose is the following. First, we fix a constant $d\lambda_{min} = 10^{-5}$, which represents the lower limit for $d\lambda$. When the PATH algorithm starts, $d\lambda$ is set to $d\lambda_{min}$. If we see after an update $\lambda_{new} = \lambda + d\lambda$ that $|F_{\lambda_{new}}(P^*(\lambda)) - F_\lambda(P^*(\lambda))| \leq \epsilon_\lambda$ then we double $d\lambda$ and keep multiplying $d\lambda$ by 2 as long as $|F_{\lambda_{new}}(P^*(\lambda)) - F_\lambda(P^*(\lambda))| \leq \epsilon_\lambda$. On the contrary, if $d\lambda$ is too large in the sense that $|F_{\lambda_{new}}(P^*(\lambda)) - F_\lambda(P^*(\lambda))| > \epsilon_\lambda$, then we divide $d\lambda$ by 2 until the criterion $|F_{\lambda_{new}}(P^*(\lambda)) - F_\lambda(P^*(\lambda))| \leq \epsilon_\lambda$ is met, or $d\lambda = d\lambda_{min}$. Once the update on $d\lambda$ is done, we run the optimization (Frank-Wolfe) for the new value $\lambda + d\lambda$. The idea behind this simple adaptation schema is to choose $d\lambda$ which keeps $|F_{\lambda_{new}}(P^*(\lambda)) - F_\lambda(P^*(\lambda))|$ just below ϵ_λ.

Stopping criterion The choice of the update criterion $|F_{\lambda_{new}}(P^*(\lambda)) - F_\lambda(P^*(\lambda))|$ is not unique. Here we check whether the function value has been changed a lot at the given point. But in fact it may be more interesting to trace the minimum of the objective function. To compare the new minimum with the current one, we need to check the distance between these minima and the difference between function values. It means that we use the following condition as the stopping criterion

$$|F_{\lambda_{new}}(P^*(\lambda_{new})) - F_\lambda(P^*(\lambda))| < \epsilon_\lambda^F \text{ and}$$
$$||P^*(\lambda_{new}) - P^*(\lambda)|| < \epsilon_\lambda^P$$

Although this approach takes a little bit more computations (we need to run Frank-Wolfe on each update of $d\lambda$), it is quite efficient if we use the adaptation

schema for $d\lambda$.

To fix the values ϵ_λ^F and ϵ_λ^P we use a parameter M which defines a ratio between these parameters and the parameters of the stopping criterion used in the Frank-Wolfe algorithm: ϵ_{FW}^F (limit value of function decrement) and ϵ_{FW}^P (limit value of argument changing): $\epsilon_\lambda^F = M\epsilon_{FW}^F$ and $\epsilon_\lambda^P = M\epsilon_{FW}^P$. The parameter M represents an authorized level of stopping criterion relaxation when we increment λ. In practice, it means that when we start to increment λ we may move away from the local minima and the extent of this move is defined by the parameter M. The larger the value of M, the further we can move away and the larger $d\lambda$ may be used. In other words, the parameter M controls the width of the tube around the path of optimal solutions.

2.3.6 Algorithm complexity

Here we present the complexity of the algorithms discussed in the paper.

- Umeyama's algorithm has three components: matrix multiplication, calculation of eigenvectors and application of the Hungarian algorithm for (2.2). Complexity of each component is equal to $O(N^3)$. Thus Umeyama's algorithm has complexity $O(N^3)$.

- LP approach (2.4) has complexity $O(N^7)$ (worst case) because it may be rewritten as an linear optimization problem with $3N^2$ variables [Boyd and Vandenberghe, 2004].

In the PATH algorithm, there are three principal parameters which have a big impact on the algorithm complexity. These parameters are ϵ_{FW}^F, ϵ_{FW}^P, M and N. The first parameter ϵ_{FW} defines the precision of the Frank-Wolfe algorithm, in some cases its speed may be sublinear [Bertsekas, 1999], however it should work much better when the optimization polytope has a "smooth" border [Bertsekas, 1999].

The influence of the ratio parameter M is more complicated. In practice, in order to ensure that the objective function takes values between 0 and 1, we usually use

the normalized version of the objective function:

$$F_{norm}(P) = \frac{||A_G P - P A_H||_F^2}{||A_G||_F^2 + ||A_H||_F^2}$$

In this case if we use the simple stopping criterion based on the value of the objective function then the number of iteration over λ (number of Frank-Wolfe algorithm runs) is at least equal to $\frac{C}{M\epsilon_{FW}^F}$ where $C = \min_{\mathcal{P}} F_{norm} - \min_{\mathcal{D}} F_{norm}$.

The most important thing is how the algorithm complexity depends on the graph size N. In general the number of iterations of the Frank-Wolfe algorithm scales as $O\left(\frac{\kappa}{\epsilon_{FW}^F}\right)$ where κ is the conditional number of the Hessian matrix describing the objective function near a local minima [Bertsekas, 1999]. It means that in terms of numbers of iterations, the parameter N is not crucial. N defines the dimensionality of the minimization problem, while κ may be close to zero or one depending on the graph structures, not explicitly on their size. On the other hand, N has a big influence on the cost of one iteration. Indeed, in each iteration step we need to calculate the gradient and to minimize a linear function over the polytope of doubly stochastic matrices. The gradient estimation and the minimization may be done in $O(N^3)$. In Section 2.4.2 we present the empirical results on how algorithm complexity and optimization precision depend on M (Figure 2.7b) and N (Figure 2.8).

2.3.7 Vertex pairwise similarities

If we match two labeled graphs, then we may increase the performance of our method by using information on pairwise similarities between their nodes. In fact one method of image matching uses only this type of information, namely shape context matching [Belongie et al., 2002]. To integrate the information on vertex similarities we use the approach proposed in (1.5), but in our case we use $F_\lambda(P)$ instead of $F_0(P)$

$$\min_P \; F_\lambda^\alpha(P) = \min_P \; (1-\alpha)F_\lambda(P) + \alpha \mathrm{tr}(C^T P), . \tag{2.20}$$

The advantage of the last formulation is that $F_\lambda^\alpha(P)$ is just $F_\lambda(P)$ with an additional linear term. Therefore we can use the same algorithm for the minimization of $F_\lambda^\alpha(P)$

as the one we presented for the minimization of $F_\lambda(P)$.

2.4 Simulations

2.4.1 Synthetic examples

In this section we compare the proposed algorithm with some classical methods on artificially generated graphs. Our choice of random graph types is based on [Newman et al., 2001] where authors discuss different types of random graphs which are the most frequently observed in various real world applications (world wide web, collaborations networks, social networks, etc...). Each type of random graphs is defined by the distribution function of node degree Prob(node degree $= k$) $= VD(k)$. The vector of node degrees of each graph is supposed to be an i.i.d sample from $VD(k)$. In our experiments we have used the following types of random graphs:

Binomial law	$VD(k) = C_N^k p^k (1-p)^{1-k}$
Geometric law	$VD(k) = (1 - e^{-\mu}) e^{\mu k}$
Power law	$VD(k) = C_\tau k^{-\tau}$

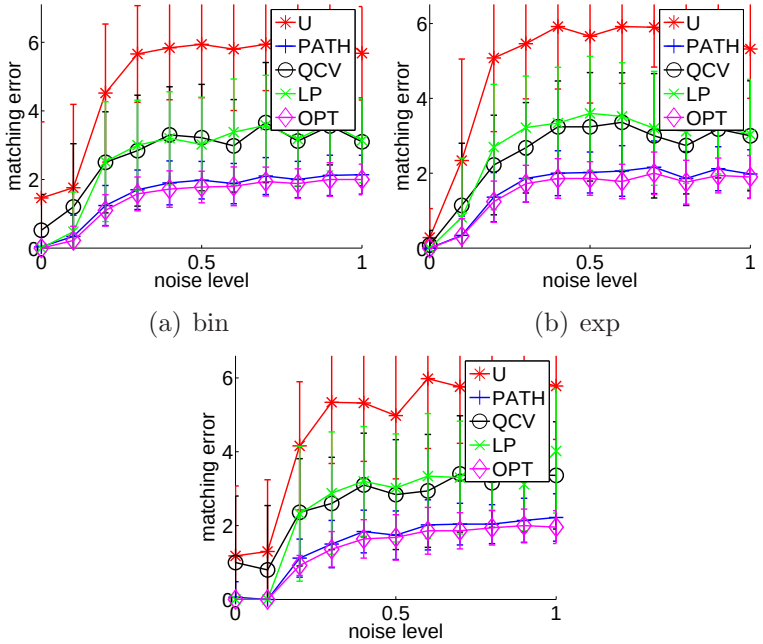
The schema of graph generation is the following

1. generate a sample $d = (d_1, \ldots, d_N)$ from $VD(k)$
2. if $\sum_i d_i$ is odd then goto step 1
3. while $\sum_i d_i > 0$
   (a) choose randomly two non-zero elements from d: d_{n1} and d_{n2}
   (b) add edge $(n1, n2)$ to the graph
   (c) $d_{n1} \leftarrow d_{n1} - 1 \quad d_{n2} \leftarrow d_{n2} - 1$

If we are interested in isomorphic graph matching then we compare just the initial graph and its randomly permuted copy. To test the matching of non-isomorphic graphs, we add randomly σN_E edges to the initial graph and to its permitted copy, where N_E is the number of edges in the original graph, and σ is the noise level.

2.4.2 Results

The first series of experiments are experiments on small size graphs (N=8), here we are interested in comparison of the PATH algorithm (see Figure 2.2), the **QCV** approach (2.7), Umeyama spectral algorithm (2.2), the linear programming approach (2.4) and exhaustive search which is feasible for the small size graphs. The algorithms were tested on the three types of random graphs (binomial, exponential and power). The results are presented in Figure 2.4. The same experiment was repeated for middle-



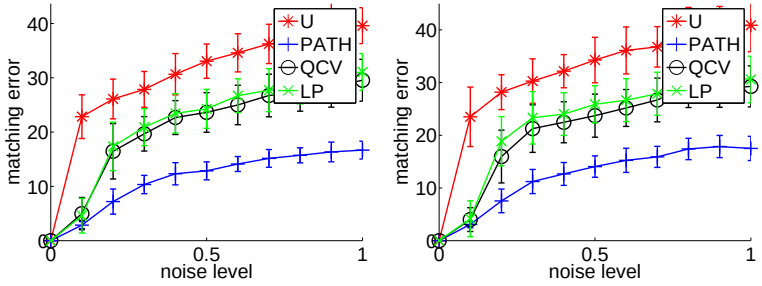
(a) bin (b) exp

(c) pow

Figure 2.4: Matching error (mean value over sample of size 100) as a function of noise. Graph size N=8. U — Umeyama's algorithm, LP — linear programming algorithm, QCV — convex optimization, PATH — path minimization algorithm,OPT — an exhaustive search (the global minimum). The range of error bars is the standard deviation of matching errors

sized graphs ($N = 20$, Figure 2.5) and for large graphs ($N = 100$, Figure 2.6).

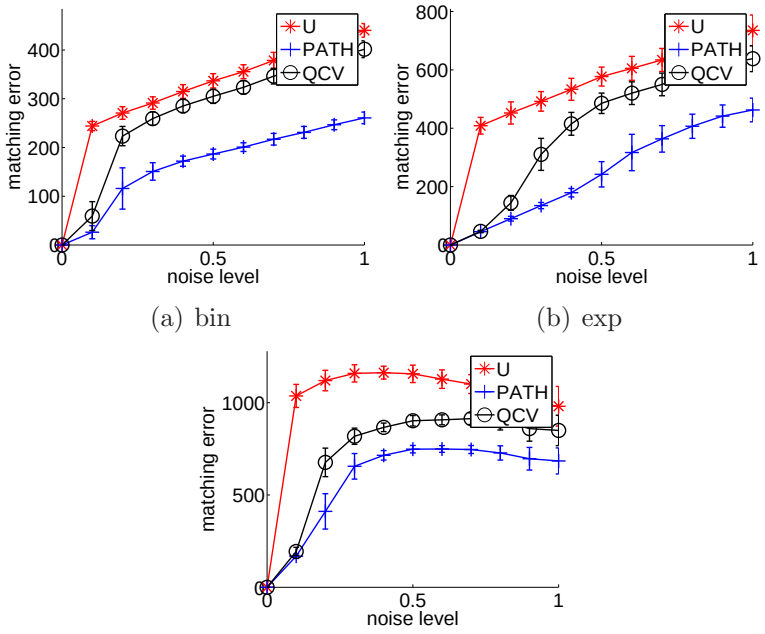
In all cases, the PATH algorithm works much better than all other approximate algorithms. There are some important things to note here. First, the choice of norm in (2.1) is not very important — results of QCV and LP are about the same. Second,



(a) bin (b) exp

(c) pow

Figure 2.5: Matching error (mean value over sample of size 100) as a function of noise. Graph size N=20. U — Umeyama's algorithm, LP — linear programming algorithm, QCV — convex optimization, PATH — path minimization algorithm.



(a) bin (b) exp

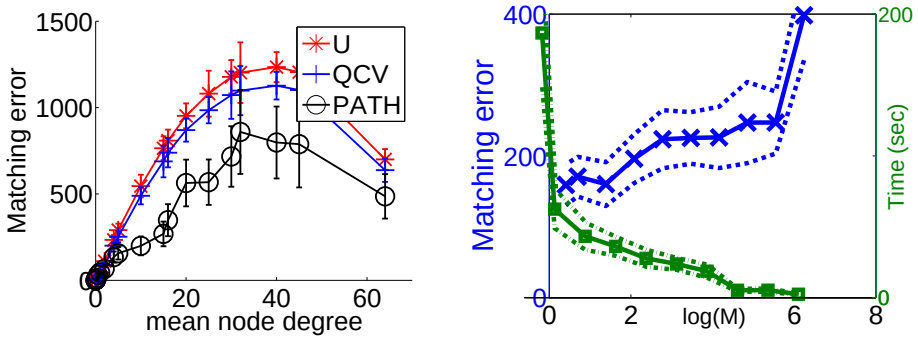
(c) pow

Figure 2.6: Matching error (mean value over sample of size 100) as a function of noise. Graph size N=100. U — Umeyama's algorithm, QCV — convex optimization, PATH — path minimization algorithm.

following the solution paths is very useful compared to just minimizing the convex relaxation and projecting the solution on the set of permutation matrices (PATH algorithms works much better than QCV). Another noteworthy observation is that the performance of PATH is very close to the optimal solution when the later can be evaluated.

We note that sometimes the matching error decreases as the noise level increases (e.g., in Figures 2.6c,2.5c), which can be explained as follows. The matching error is upper bounded by the minimum of the total number of zeros in the adjacency matrices A_G and A_H, so in general this upper bound deacreases when the edge density increases. When the noise level increases, it makes graphs denser, and consequently the upper bound of matching error decreases. The general behavior of graph matching algorithms as functions of the graph density is presented in Figure 2.7a). Here again the matching error decreases when the graph density becomes very large.

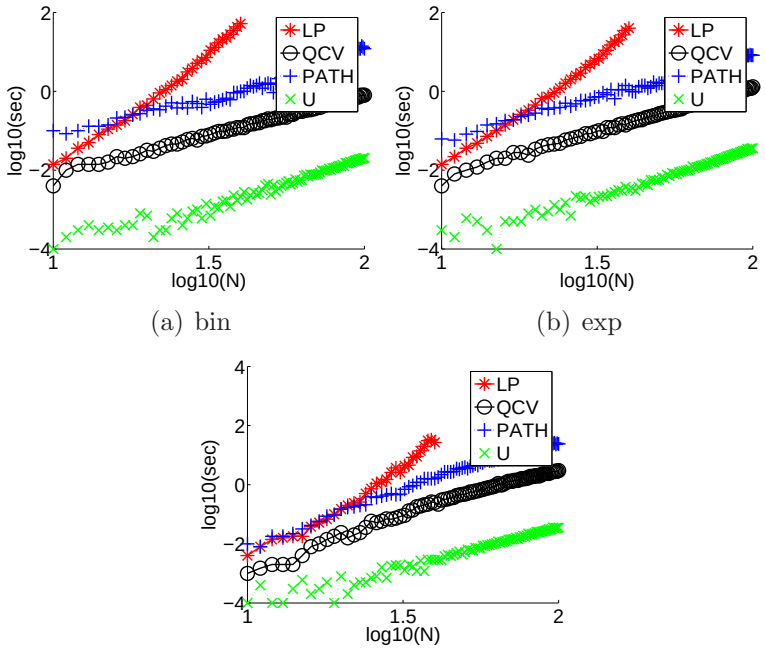
The parameter M (see section 2.3.5) defines how precisely the PATH algorithm tries to follow the path of local minimas. The larger M, the faster the PATH algorithm. At the extreme, when M is close to $1/\epsilon_{FW}$, we jump directly from the convex function ($\lambda = 0$) to the concave one ($\lambda = 1$). Figure 2.7b) shows in more details how algorithm speed and precision depend on M.



(a) (b)

Figure 2.7: (a) Algorithm performance as a function of graph density. (b) Precision and speed of the PATH algorithm as a function of M, the relaxation constant used in the PATH algorithm (see section 2.3.5). In both cases, graph size N=100, noise level σ=0.3, sample size is equal to 30. Error bars represent standard deviation of the matching error (not averaged)

Another important aspect to compare the different algorithms is their run-time complexity as a function of N. Figure 2.8 shows the time needed to obtain the matching between two graphs as a function of the number of vertices N (for N between 10 and 100), for the different methods. These curves are coherent with theoretical values of algorithm complexities summarized in Section 2.3.6. In particular we observe that Umeyama's algorithm is the fastest method, but that QCV and PATH have the same complexity in N. The LP method is competitive with QCV and PATH for small graphs, but has a worse complexity in N.



(a) bin (b) exp

(c) pow

Figure 2.8: Timing of U,LP,QCV and PATH algorithms as a function of graph size, for the different random graph models. LP slope ≈ 6.7, U, QCV and PATH slope ≈ 3.4

2.5 QAP benchmark library

The problem of graph matching may be considered as a particular case of the quadratic assignment problem (QAP). The minimization of the loss function (2.1) is equivalent

Table 2.1: Experiment results for QAPLIB benchmark datasets.

QAP	MIN	PATH	QPB	GRAD	U
chr12c	11156	**18048**	20306	19014	40370
chr15a	9896	**19086**	26132	30370	60986
chr15c	9504	**16206**	29862	23686	76318
chr20b	2298	**5560**	6674	6290	10022
chr22b	6194	**8500**	9942	9658	13118
esc16b	292	300	**296**	298	306
rou12	235528	**256320**	278834	273438	295752
rou15	354210	391270	**381016**	457908	480352
rou20	725522	**778284**	804676	840120	905246
tai10a	135028	**152534**	165364	168096	189852
tai15a	388214	**419224**	455778	451164	483596
tai17a	491812	**530978**	550852	589814	620964
tai20a	703482	**753712**	799790	871480	915144
tai30a	1818146	**1903872**	1996442	2077958	2213846
tai35a	2422002	**2555110**	2720986	2803456	2925390
tai40a	3139370	**3281830**	3529402	3668044	3727478

to the maximization of the following function:

$$\max_P \ \mathrm{tr}(P^T A_G^T P A_H)\,.$$

Therefore it is interesting to compare our method with other approximate methods proposed for QAP. Schellewald et al. [2001] proposed the QPB algorithm for that purpose and tested it on matrices from the QAP benchmark library [Cela, 2007], QPB results were compared to the results of graduated assignment algorithm GRAD[Gold and Rangarajan, 1996] and Umeyama's algorithm. Results of PATH application to the same matrices are presented in Table 2.1, scores for QPB and graduated assignment algorithm are taken directly from the publication [Schellewald et al., 2001]. We observe that on 14 out of 16 benchmark, PATH is the best optimization method among the methods tested.

2.6 Image processing

In this section, we present two applications in image processing. The first one (Section 2.6.1) illustrates how taking into account information on graph structure may increase image alignment quality. The second one (Section 2.6.2) shows that the structure of contour graphs may be very important in classification tasks. In both examples we compare the performance of our method with the shape context approach [Belongie et al., 2002], a state-of-the-art method for image matching.

2.6.1 Alignment of vessel images

The first example is dedicated to the problem of image alignment. We consider two photos of vessels in human eyes. The original photos and the images of extracted vessel contours (obtained from the method of Walter et al. [2003]) are presented in Figure 2.9. To align the vessel images, the shape context algorithm uses the context radial histograms of contour points [Belongie et al., 2002]. In other words, according to the shape context algorithm one aligns points which have similar context histograms. The PATH algorithm uses also information about the graph structure. When we use the PATH algorithm we have to tune the parameter α (2.20), we tested several possible values and we took the one which produced the best result. To construct graph we use all points of vessel contours as graph nodes and we connect all nodes within a circle of radius r (in our case we use $r = 50$). Finally, to each edge (i, j) we associate the weight $w_{i,j} = \exp(-|x_i - y_j|)$.

A graph matching algorithm produces an alignment of image contours, then to align two images we have to expand this alignment to the rest of image. For this purpose, we use a smooth spline-based transformation [Bookstein, 1989]. In other words, we estimate parameters of the spline transformation from the known alignment of contour points and then we apply this transformation to the whole image. Results of image matching based on shape context algorithm and on PATH algorithm are presented in Figure 2.10, where black lines designate connections between associated points. We observe that the context shape method creates many unwanted matching, while PATH produces a matching that visually corresponds to a correct alignment

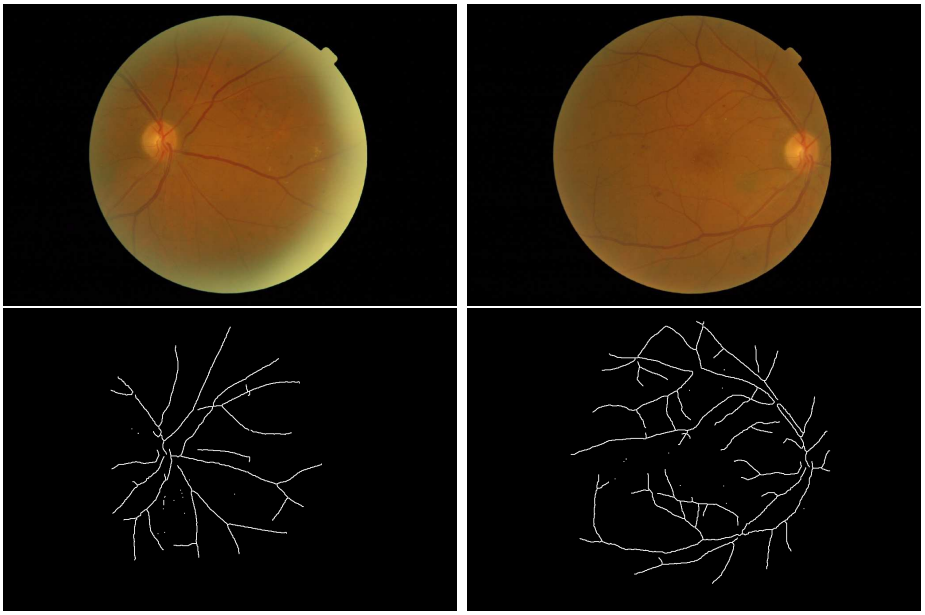


Figure 2.9: Eye photos (top) and vessel contour extraction (bottom).

of the structure of vessels. The main reason why graph matching works better than shape context matching is the fact that shape context does not take into account the relational positions of matched points and may lead to totally incoherent graph structures. In contrast, graph matching tries to match pairs of nearest points in one image to pairs of nearest points in another one.

Among graph matching methods, different results are obtained with different optimization algorithms. Table 2.2 shows the matching errors produced by different algorithms on this vessel alignment problem. The PATH algorithm has the smallest matching error, with the alignment shown on Figure 2.10. QCV comes next, with an alignment that is also visually correct. On the other hand, the Umeyama algorithm has a much larger matching error, and visually fails to find a correct alignment, similar to the shape context method.

Table 2.2: Alignment of vessel images, algorithm performance

Method	Shape context	Umeyama	QCV	PATH
matching error	870.61	764.83	654.42	625.75

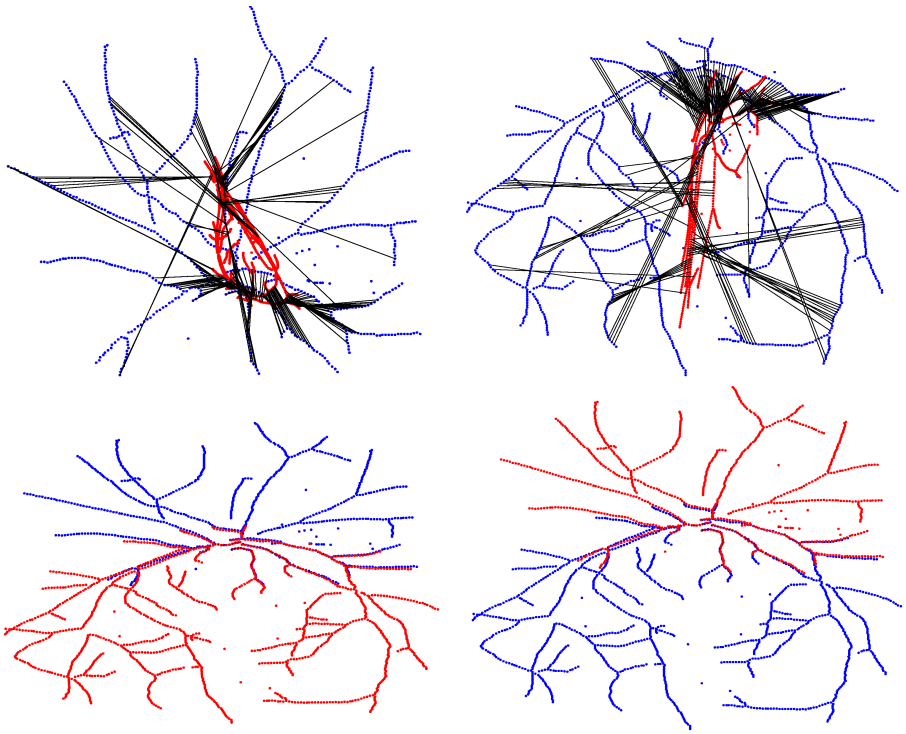


Figure 2.10: Comparison of alignment based on shape context (top) and alignment based on the PATH optimization algorithm (bottom). For each algorithm we present two alignments: image '1' on image '2' and the inverse. Each alignment is a spline-based transformation (see text).

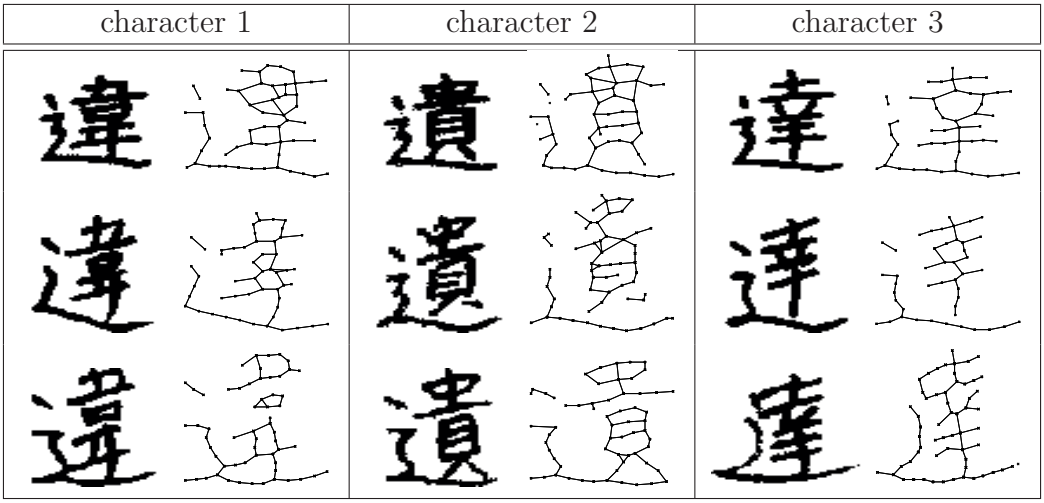
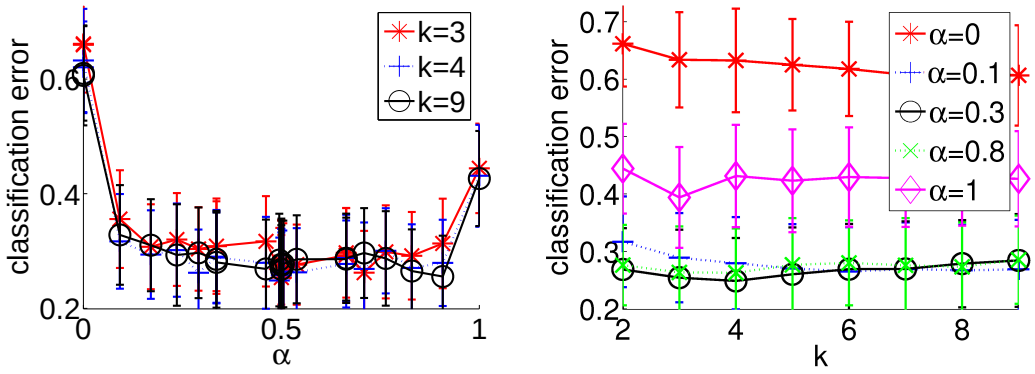


Figure 2.11: Chinese characters from the ETL9B dataset.

2.6.2 Recognition of handwritten chinese characters

Another example that we consider in this chapter is the problem of chinese character recognition from the Saito1985ETL9b dataset [Saito et al., 1985]. The main idea is to use a score of optimal matching as a similarity measure between two images of characters. This similarity measure can be used then in machine learning algorithms, K-nearest neighbors (KNN) for instance, for character classification. Here we compare the performance of four methods: linear support vector machine (SVM), SVM with gaussian kernel, KNN based on score of shape context matching and KNN based on scores from graph matching which combines structural and shape context information. As a score, we use just the value of the objective function (2.20) at the (locally) optimal point. We have selected three chinese characters known to be difficult to distinguish by automatic methods. Examples of these characters as well as examples of extracted graphs (obtained by thinning and uniformly subsampling the images) are presented in Figure 2.11. For SVM based algorithms, we use directly the values of image pixels (so each image is represented by a binary vector), in graph matching algorithm we use binary adjacency matrices of extracted graphs and shape context matrices [Belongie et al., 2002].

Our data set consist of 50 exemples (images) of each class. Each image is represented by 63×64 binary matrix. To compare different methods we use the cross validation error (five folds). The dependency of classification error from two algorithm parameters (α — coefficient of linear combination (2.20) and k — number of nearest neighbors used in KNN)) is shown in Figure 2.12.



(a) (b)

Figure 2.12: (a) Classification error as a function of α. (b) Classification error as a function of k. Classification error is estimated as cross validation error (five folds, 50 repetition), the range of the error bars is the standard deviation of test error over one fold (not averaged over folds and repetition)

Two extreme choices $\alpha = 1$ and $\alpha = 0$ correspond respectively to pure shape context matching, i.e., when only node labels information is used, and pure unlabeled graph matching. It is worth observing here that KNN based just on the score of unlabeled graph matching does not work very well, the classification error being about 60%. An explanation of this phenomenon is the fact that learning patterns have very unstable graph structure within one class. The pure shape context method has a classification error of about 39%. The combination of shape context and graph structure informations allows to decrease the classification error down to 25%. Beside the PATH algorithm, we tested also the QCV algorithm and the Umeyama algorithm,the Umeyama algorithm almost does not decrease the classification error. The QCV algorithm works better than then Umeyama algorithm, but still worse than the PATH algorithm. Complete results can be found in Table 2.3.

Table 2.3: Classification of chinese characters. (CV, STD)—mean and standard deviation of test error over cross-validation runs (five folds, 50 repetitions)

Method	CV	STD
Linear SVM	0.377	± 0.090
SVM with gaussian kernel	0.359	± 0.076
KNN (PATH) (α=1): shape context	0.399	± 0.081
KNN (PATH) (α=0.4)	0.248	± 0.075
KNN (PATH) (α=0): pure graph matching	0.607	± 0.072
KNN (U) (α=0.9): α best choice	0.382	± 0.077
KNN (QCV) (α=0.3): α best choice	0.295	± 0.061

2.7 Conclusion

We have presented the PATH algorithm, a new technique for graph matching based on convex-concave relaxations of the initial integer programming problem. PATH allows to integrate the alignment of graph structural elements with the matching of vertices with similar labels. Its results are competitive with state-of-the-art methods in several graph matching and QAP benchmark experiments. Moreover, PATH has a theoretical and empirical complexity competitive with the fastest available graph matching algorithms.

Two points can be mentioned as interesting directions for further research. First, the quality of the convex-concave approximation is defined by the choice of convex and concave relaxation functions. Better performances may be achieved by more appropriate choices of these functions. Second, another interesting point concerns the construction of a good concave relaxation for the problem of directed graph matching[5], i.e., for asymmetric adjacency matrix. Such generalizations would be interesting also as possible polynomial-time approximate solutions for the general QAP problem.

2.A A toy example

The PATH algorithm does not generally find the global optimum of the NP-complete optimization problem. In this appendix we illustrate with two examples how the set

[5]A possible solution is to use the transformation procedure described in Preface.

of local optima tracked by PATH may or may not lead to the global optimum.

More precisely, we consider two simple graphs with the following adjacency matrices:

$$\mathbf{G} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix} \text{ and } \mathbf{H} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}.$$

Let C denote the cost matrix of vertex association

$$\mathbf{C} = \begin{bmatrix} 0.1691 & 0.0364 & 1.0509 \\ 0.6288 & 0.5879 & 0.8231 \\ 0.8826 & 0.5483 & 0.6100 \end{bmatrix}.$$

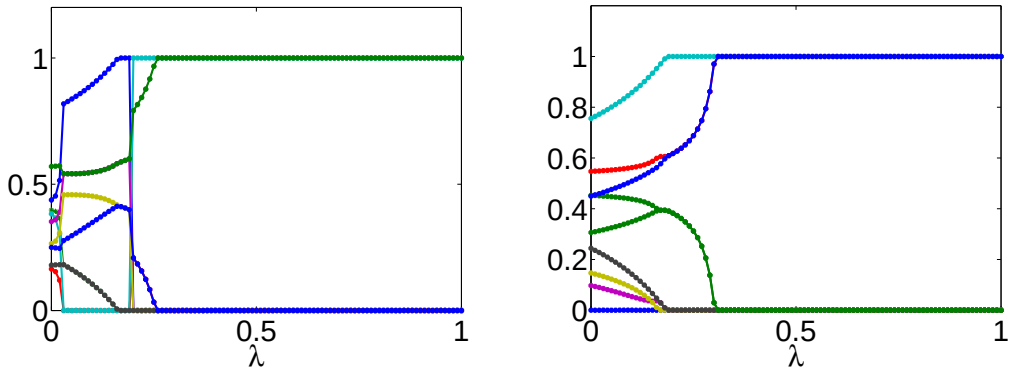
Let us suppose that we have fixed the tradeoff $\alpha = 0.5$, and that our objective is then to find the global minimum of the following function:

$$F_0(P) = 0.5||GP - PH||_F^2 + 0.5\text{tr}(C'P), \qquad P \in \mathcal{P}. \tag{2.21}$$

As explained earlier, the main idea underlying the PATH algorithm is to try to follow the path of global minima of $F_\lambda^\alpha(P)$ (2.20). This may be possible if all global minima P_λ^* form a continuous path, which is not true in general. In the case of small graphs we can find the exact global minimum of $F_\lambda^\alpha(P)$ for all λ. The trace of global minima as functions of λ is presented in Figure 2.13(a) (i.e., we plot the values of the nine parameters of the doubly stochastic matrix, which are, as expected, all equal to zero or one when $\lambda = 1$). When λ is near 0.2 there is a jump of global minimum from one face to another. However if we change the linear term C to

$$\mathbf{C}' = \begin{bmatrix} 0.4376 & 0.3827 & 0.1798 \\ 0.3979 & 0.3520 & 0.2500 \\ 0.1645 & 0.2653 & 0.5702 \end{bmatrix},$$

then the trace becomes smooth (see Figure 2.13(b)) and the PATH algorithm then finds the globally optimum point. Characterizing cases where the path is indeed smooth is the subject of ongoing research.



(a) (b)

Figure 2.13: Nine coordinates of global minimum of F_λ^α as a function of λ

2.B Kronecker product

The Kronecker product of two matrices $A \otimes B$ is defined as follows:

$$A \otimes B = \begin{bmatrix} Ba_{11} & \cdots & Ba_{1n} \\ \vdots & \ddots & \vdots \\ Ba_{m1} & \cdots & Ba_{mn} \end{bmatrix}.$$

Two important properties of Kronecker product that we use in this chapter are:

$$(A^T \otimes B)\mathrm{vec}(X) = \mathrm{vec}(BXA),$$
$$\text{and } \mathrm{tr}(X^T A X B^T) = \mathrm{vec}(X)^T (B \otimes A)\mathrm{vec}(X)\,.$$

Chapter 3

Global alignment of protein-protein interaction networks by graph matching methods

Preface

Aligning protein-protein interaction (PPI) networks of different species has drawn a considerable interest recently. This problem is important to investigate evolutionary conserved pathways or protein complexes across species, and to help in the identification of functional orthologs through the detection of conserved interactions. It is however a difficult combinatorial problem, for which only heuristic methods have been proposed so far.

We reformulate the PPI alignment as a graph matching problem, and investigate how state-of-the-art graph matching algorithms can be used for that purpose. We differentiate between two alignment problems, depending on whether strict constraints on protein matches are given, based on sequence similarity, or whether the goal is instead to find an optimal compromise between sequence similarity and interaction conservation in the alignment. We propose new methods for both cases, and assess their performance on the alignment of the yeast and fly PPI networks. The new methods consistently outperform state-of-the-art algorithms, retrieving in particular

78% more conserved interactions than IsoRank for a given level of sequence similarity.

This chapter is a slightly modified version of [Zaslavskiyi et al., 2009].

3.1 Introduction

PPIs play a central role in most biological processes. Recent years have witnessed impressive progress towards the elucidation of large-scale PPI networks in various organisms, thanks in particular to the development of high-throughput experimental techniques such as yeast two-hybrid [Fields and Song, 1989] or coimmunoprecipitation followed by mass-spectrometry [Aebersold and Mann, 2003]. As the amount of PPI network data increases, computational methods to analyze and compare them are also being developed at a fast pace. In particular, comparative PPI network analysis across species has already provided insightful views of similarities and differences between species at the systemic level [Sharan et al., 2005; Suthram et al., 2005] and helped in the identification of functional orthologs [Bandyopadhyay et al., 2006].

Comparing PPI networks usually involves some form of *network alignment*, i.e., the identification of pairs of homologous proteins from two different organisms, such that PPIs are conserved between matched pairs. The rationale behind this notion is that a protein and its functional orthologs are likely to interact with proteins in their respective network that are themselves functional orthologs. Hence, while direct sequence homology alone is often not sufficient to identify functional orthologs within paralogous families [Sjölander, 2004], the use of PPI information can help in the disambiguation of functional orthologs within clusters of homologous sequences, such as those produced by the Inparanoid algorithm [Remm et al., 2001]. This approach has been investigated in particular by Bandyopadhyay et al. [2006]. Conversely, network alignment can also be a valuable approach to validate PPI conserved across multiple species and detect evolutionary conserved pathways or protein complexes [Sharan et al., 2005; Kelley et al., 2003].

Several methods have been proposed to perform *local* network alignment (LNA) of PPI networks, i.e., to identify subsets of matching pairs of proteins with conserved subgraphs of interactions. These methods include PathBLAST [Kelley et al.,

2003, 2004] and NetworkBLAST [Sharan et al., 2005], which adapt the ideas of the BLAST algorithm to the search for local alignments between graphs, the method of Koyutürk et al. [2006], inspired by biological models of deletion and duplication, Graemlin [Flannick et al., 2006], which uses networks of modules to infer the alignment, or the Bayesian approach of Berg and Lässig [2006]. Less attention has been paid to the problem of *global* network alignment (GNA), i.e., the search for a global correspondence between most or all vertices of two networks which again matches similar proteins and leads to conserved interactions. Notable exceptions include the Markov random field (MRF) based method of Bandyopadhyay et al. [2006] and the IsoRank algorithm [Singh et al., 2008] which formulates the problem as an eigenvalue problem.

While LNA procedures can detect multiple, unrelated matched regions between networks, and can in particular match a given protein of a network to several proteins of the other network in different local matchings, GNA seeks the best consistent matching across all nodes simultaneously. This can be a desirable property for many applications, such as functional ortholog identification. On the other hand, from a computational point of view, GNA is arguably more difficult than LNA since it must find a solution among all possible global matchings. In fact, as we explain below, it is natural to reformulate GNA as weighted graph matching problem, a problem for which no polynomial-time algorithm is known. Solving the general GNA problem therefore must involve some sort of approximate or heuristic method, such as IsoRank.

Following this line of thought, we propose here to formulate explicitly GNA as a graph matching problem, and investigate the use of modern state-of-the-art exact and approximate methods to solve it. While no exact solution of the graph matching optimization problem can be found in general, we show that in certain cases, if "enough constraints" are put on the possible protein associations, and if the PPI networks are "not too dense" (these notions being rigorously defined in Section 3.3.2), then an exact solution can be found efficiently by a new message-passing algorithm. Interestingly, this case arises in particular in the functional ortholog detection problem between yeast and fly investigated by Bandyopadhyay et al. [2006], where matching pairs are constrained to belong to clusters of proteins produced by the Inparanoid

algorithm and the PPI networks of both species are not too dense. On these data, we are therefore able to find a matching which conserves more interactions than the solutions found by MRF [Bandyopadhyay et al., 2006] as well as a version of IsoRank adapted to this situation [Singh et al., 2008], and we are in fact certain that our solution is optimal in the sense that it produces the largest possible number of conserved interactions. Interestingly, the resulting alignment retrieves 13% more HomoloGene pairs than the alignments of MRF and 5% more than that of IsoRank, suggesting that maximizing the number of conserved interactions indeed improves functional orthology disambiguation. When the GNA is more complex, e.g., matched pairs are not limited to belong to the same Inparanoid clusters, or the PPI networks have more edges, then our message-passing algorithm can not be used and the optimal matching can not be found in reasonable time anymore. In that case we propose to use a recent state-of-the-art approximate methods for graph matching [Zaslavskiy et al., 2008b], which tracks a path of solutions for a family of relaxed problems, as well as a new, faster and more direct gradient-based method, which bears similarities with the IsoRank method. Like IsoRank, these methods have a free parameter to balance the trade-off between matching similar proteins, on the one hand, and producing an alignment with many conserved interactions, on the other hand. We test them on the global unconstrained alignment of the fly and yeast networks, and show that for a given level of mean sequence similarity between matched proteins, our new method retrieves 78% more conserved interactions than IsoRank.

3.2 Constrained and balanced GNA problems

In this section we set the notations and formalize two variants of the GNA problems. We represent a PPI network describing the interactions among N proteins of an organism as an undirected simple graph $G = (V_G, E_G)$, where $V_G = (v_1, \ldots, v_N)$ is a finite set of N vertices representing the N proteins, and $E_G \subset V_G \times V_G$ is the set of edges representing the pairs of interacting proteins. Each such graph (or network) can equivalently be represented by a symmetric $N \times N$ adjacency matrix A_G where $[A_G]_{ij} = [A_G]_{ji} = 1$ if protein v_i interacts with protein v_j and 0 otherwise.

Given two graphs G and H representing the PPI networks of two species, the GNA problem is, roughly speaking, to find a correspondence between the vertices of G and the vertices of H which matches similar proteins and enforces as much as possible the conservation of interactions between matched pairs in the two graphs. To formalize this, let us assume that G and H have the same number N of vertices, and that we are looking for a bijection between the vertices of G and the vertices of H. Although this may sound at first sight a strong assumption, given that PPI networks usually do not have the same size, and that we may not want to match all proteins of each network, both limitations can be addressed by adding dummy nodes (with no connection) to each graph in order to ensure that they finally have the same size. In a complete matching of such graphs with dummy nodes, matching a protein to a dummy node simply means that in the GNA the protein is not matched. G and H being assumed to have the same number of vertices, a matching of their vertices is now simply a permutation π of $\{1, \ldots, N\}$ which associates the i-th vertex of H with the $\pi(i)$-th vertex of G. Equivalently, the permutation π can be represented by a $N \times N$ permutation matrix P, i.e., a binary matrix whose (i, j)-th entry is equal to 1 if and only if $\pi(i) = j$ (that is, when the i-th vertex of H is matched to the j-th vertex of G). We denote by $\mathcal{P} = \{P \in \{0, 1\}^{N \times N} : P1_N = 1_N, P^T 1_N = 1_N\}$ the set of permutation matrices, where 1_N is the N-dimensional vectors whose entries are all equal to 1.

The number of interactions conserved by a permutation π is the number of pairs (i, j) which are connected in H, and such that their corresponding vertices $\pi(i)$ and $\pi(j)$ are also connected in G. Let us denote by $J(P)$ the number of such interactions conserved by the permutation encoded in the permutation matrix P. In order to express $J(P)$, we can observe that if we apply the permutation encoded by P to the vertices of H, we obtain a new graph isomorphic to H which we denote by $P(H)$. It is easy to see that the adjacency matrix of the permuted graph, $A_{P(H)}$, is simply obtained from A_H by the equality $A_{P(H)} = PA_HP^T$ [Umeyama, 1988]. As a result, $J(P)$ is simply obtained as half the number of entries which are simultaneously equal to 1 in both binary matrices A_G and PA_HP^T (each conserved interaction results in two identical entries, by symmetry of the adjacency matrices). Hence we obtain the

following expression for $J(P)$:

$$J(P) = \frac{1}{2}\sum_{i,j=1}^{N}[A_G]_{ij}[PA_HP^T]_{ij} = \frac{1}{2}\mathrm{tr}(A_G^TPA_HP^T)\,. \tag{3.1}$$

Besides the number of conserved interactions, a good GNA should match proteins with similar sequences. We consider here two possible formulations of this objective.

- *Constrained GNA*. Here we assume that a pre-processing of the protein sequences has produced a set of candidate matched pairs $\mathcal{A} \subset V_H \times V_G$, and we simply wish to disambiguate the matching using PPI information, if some proteins have several candidate matchings. This is for example the formulation proposed by Bandyopadhyay et al. [2006], where a first clustering of all proteins sequences is performed to define a collection of protein clusters with the Inparanoid algorithm, and the pairs matched between the yeast and fly proteome are constrained to belong to the same cluster. Such constraints can be directly encoded as constraints over the permutation matrix P, by imposing $P_{ij} = 0$ if the i-th vertex of the first graph and the j-th vertex of the second graph are not allowed to match. We are then looking for a solution in the set of matrices $\mathcal{P}_\mathcal{A} = \{P \in \mathcal{P} : \forall (i,j) \in [1,N]^2 \backslash \mathcal{A}, P_{ij} = 0\}$, and it is then natural to look for the permutation compatible with the constraints with the largest number of conserved interactions, i.e., to solve:

$$\max_{P\in\mathcal{P}_\mathcal{A}} J(P)\,. \tag{3.2}$$

- *Balanced GNA*. A interesting property of constrained GNA is that, by reducing the search space to $\mathcal{P}_\mathcal{A}$, it can result in a tractable optimization problem (as shown for example in Section 3.3.2). On the other hand, in some cases one may want to accept matching between less similar vertices if it leads to an important increase in the number of conserved interactions. In other words, one would like to be able to automatically *balance* the matching of similar vertices with the conservation of interactions, as advocated by Singh et al. [2008] and

implemented by IsoRank. This can be formalized by assuming that a $N \times N$ matrix of similarities between vertices C is given (e.g., derived from pairwise sequence similarity scores), and by trying to maximize the total similarity between matched pair. C_{ij} denoting the similarity between the i-th vertex of G and the j-th vertex of H, the total similarity between pairs matched by a permutation matrix π is simply

$$S(P) = \sum_{i=1}^{N} C_{\pi(i),i} = \mathrm{tr}\,(PC)\ . \tag{3.3}$$

In order to find a balance between matching similar pairs (large $S(P)$) and having many conserved interactions (large $J(P)$), we propose to consider the following optimization problem:

$$\max_{P \in \mathcal{P}} \lambda J(P) + (1-\lambda) S(P)\,, \tag{3.4}$$

where $\lambda \in [0,1]$ controls the trade-off between both objectives. $\lambda = 1$ corresponds to the maximization of $J(P)$ only, i.e., to find a good topological matching of the graphs independently of the similarity between matched pairs, while $\lambda = 0$ amounts to focus only on the similarity between proteins and finding a matching which maximized the mean sequence similarity, without using PPI information.

When $\lambda > 0$, the balanced GNA problem (3.4) is equivalent to a general graph matching problem, discussed in Section 3.3.1, which is known to be computationally intractable in general. The constrained GNA (3.2) can be seen as a particular case of the balanced GNA, by taking the similarity function equal to 0 between two vertices allowed to match and $-\infty$ for two vertices not allowed to match. Indeed, in that case (3.4) is equivalent to minimizing $J(P)$ over the set of matrices P for which $S(P)$ is finite, that is exactly the set $\mathcal{P}_{\mathcal{A}}$ of (3.2). While indeed general graph matching methods to solve (3.4) can be applied to solve (3.2), we show in the next Section that in some cases there exists a simple polynomial-time algorithm to solve (3.2) directly even for large non-sparse graphs.

3.3 Methods

In this section we present methods to solve both the constrained GNA problem (3.2) and the balanced GNA problem (3.4). Since any algorithm to solve the balanced GNA problem can also solve the constrained GNA, as explained in the previous section, we start by describing methods to solve the balanced GNA problem.

3.3.1 Algorithms for the balanced GNA problem

The balanced GNA problem (3.4) is a general graph matching problem, which is known to be a difficult combinatorial problem. While some methods based on incomplete enumeration may be applied to search for an exact optimal solution in the case of small or sparse graphs, only approximate algorithms that usually find non-optimal solutions but are more scalable can be used for large non-sparse graph matching. Many such approximate algorithms have been proposed, see e.g., the review of Conte et al. [2004]. They include in particular spectral methods [Umeyama, 1988; Caelli and Kosinov, 2004; Singh et al., 2008], or methods based on a relaxation of the optimization problem (3.4) [Almohamad and Duffuaa, 1993; Gold and Rangarajan, 1996]. They differ mainly on their scalability, and on the accuracy of the solution found. For example, a comparison of several such methods was carried out recently in [Zaslavskiy et al., 2008c,b].

Based on these observation, we propose here to use state-of-the-art graph matching methods to balanced GNA for PPI networks. In particular we focus on the PATH algorithm [Zaslavskiy et al., 2008b], which was shown to provide state-of-the-art performance in various graph matching benchmark. We also propose a new and simpler gradient ascent method, similar in spirit to the Graduated Assignment (GA) algorithm [Gold and Rangarajan, 1996]. As a benchmark, we consider the IsoRank method, which can be thought of as a particular spectral method for graph alignment, and which is currently the method of choice for balanced GNA of PPI networks. We now briefly describe these methods.

- *PATH method.* The PATH algorithm is based on two relaxations of (3.4), one concave and one convex, over the set of doubly stochastic matrices [Zaslavskiy et al.,

2008b]. The method starts by solving the convex relaxation, and then iteratively solves a linear combination of the convex and concave relaxations by gradually increasing the weight of the concave relaxation and following the path of solutions thus created. It finishes when the a solution reaches a corner of the set of doubly stochastic matrices, i.e., when the solution is a permutation matrix in $\mathcal{P}$. On several benchmarks, the PATH method was shown to be state-of-the-art in accuracy, and can easily process graphs with a few thousands vertices in a few hours on a modern desktop computer.

- *GA method.* We propose a new, simple gradient method based on a relaxation of (3.4) over the set of doubly stochastic matrices. Although the function to be maximized is not concave (because of the term $J(P)$), we simply start from an initial solution and iteratively choose a new permutation matrix in the direction of the gradient of the objective function. This approach may be relevant if we can start from a "good" initial solution, i.e., if we solve a constrained GNA (3.2) where the constraints are strong enough. The gradient of $S(P)$ in (3.3) is equal to S, the gradient of $J(P)$ in (3.1) at a matrix P_n is equal to $A_G^T P_n A_H$. Hence we propose to iteratively update the permutation matrix following the rule $P_{n+1} \leftarrow \arg\max_{P\in\mathcal{P}} \operatorname{tr}\left([\lambda A_G^T P_n A_H + (1-\lambda)C]P\right)$, which can be found efficiently by the Hungarian algorithm [Kuhn, 1955].

- *IsoRank method.* The idea of the IsoRank algorithm is to use the following recursive formula [Singh et al., 2008]

$$R(i,j) = \sum_{v\in N(i)} \sum_{u\in N(j)} \frac{1}{|N(u)||N(v)|} R(u,v),\ i \in V_G,\ j \in V_H, \qquad (3.5)$$

where $N(i)$ denotes the set of neighbors of i, V_G denotes the set of vertices of graph G and element $R(i,j)$ represents the similarity between vertex i of graph G and vertex j of graph H. In the case of PPI networks it represents the "likelihood" that proteins i and j are functional orthologs. The recursive formula says that the more i and j have similar neighbors, the greater is the similarity measure between i and j. To estimate R, Singh et al. [2008] propose

to use the power method to iteratively update R according to:

$$R \leftarrow AR/||AR||\,, \tag{3.6}$$

where A is the $N^2 \times N^2$ matrix defined as:

$$A(i,j)(u,v) = \frac{1}{|N(u)||N(v)|}\,.$$

To take into account the information on protein sequence similarities encoded by matrix C, the following modification of (3.5) is used

$$R = \lambda AR + (1-\lambda)C, \tag{3.7}$$

where λ has the same interpretation as in (3.4).

3.3.2 Algorithms for the constrained GNA problem

As explained in Section 3.2, all methods for solving the balanced GNA problem (3.4) can also be used to solve the constrained GNA problem (3.2), by using a particular similarity function to enforce the constraints. Hence a first series of methods to solve (3.2) are the constrained version of IsoRank, GA and PATH, described in the previous section. In addition to these three methods, we consider two additional approaches specifically dedicated to the constrained GNA problem: the Markov random field (MRF) method of Bandyopadhyay et al. [2006], and a new method based on message passing (MP) which we propose to find the global optimum of (3.2) when the graphs are not too dense.

- *MRF method.* To solve ambiguous assignments in Inparanoid clusters with more than two proteins, Bandyopadhyay et al. [2006] propose to use the information on protein interactions, by choosing the assignments which maximize the number of conserved interactions between two species. For that purpose they use the following probabilistic model. They associate a binary variable z_{ij} to each

possible protein ortholog pair (f_i, y_j) (here f_i and y_j denote Fly and Yeast proteins from the same Inparanoid cluster), where $z_{ij} = 1$ means that f_i and y_j are functional orthologs. Two variables z_{ij} and z_{kt} are connected if at least one pair of proteins (f_i, f_k) or (y_j, y_t) is connected in its PPI network, and the other one has a common neighbor (or is also connected).Let $N(ij)$ denote the set of indices connected to z_{ij}. Then the probability law of z_{ij} is modeled by:

$$P(z_{ij}|z_{N(ij)}) = \frac{1}{1 + exp\{-\alpha - \beta \sum_{kt \in N(ij)} z_{kt}\}} . \tag{3.8}$$

The interpretation of this formula is that z_{ij} has more chances to be equal to one when the number of neighbors equal to one is large. When there are only two proteins in cluster f_i and y_j then by definition $z_{ij} = 1$. If f_i and y_j are from different clusters then also by definition $z_{ij} = 0$. The parameters α and β are estimated on the basis of training data, then a Gibbs sampling is performed to define the value of unknown variables z on the test set. We refer to [Bandyopadhyay et al., 2006] for more details on this method.

- *MP method for exact optimization.* Although intractable in general, we now show that constrained GNA problem (3.2) can be solved exactly and efficiently in some cases, and propose a new, efficient algorithm based on message passing for that purpose. More precisely, we consider the situation where the set of proteins have been clustered into a finite set of L groups $c_1, \ldots, c_L$, which form a partition of $V_G \cup V_H$, and where only proteins within the same group can be matched[1]. This situation, illustrated in Figure 3.1, represents for example the problem investigated by Bandyopadhyay et al. [2006], where proteins of two organisms are first clustered by the Inparanoid algorithm, and functional orthologs are searched within clusters. Let us now consider the L clusters as vertices of a graph, and connect two clusters c_i and c_j if they contain proteins of both organisms that interact in their respective PPI network. For example, in Figure 3.1, c_1 and c_2 are connected because c_1 contains f_1 from the first organism

[1]Technically, we add dummy nodes in each cluster to obtain the same number of proteins of each species in each cluster.

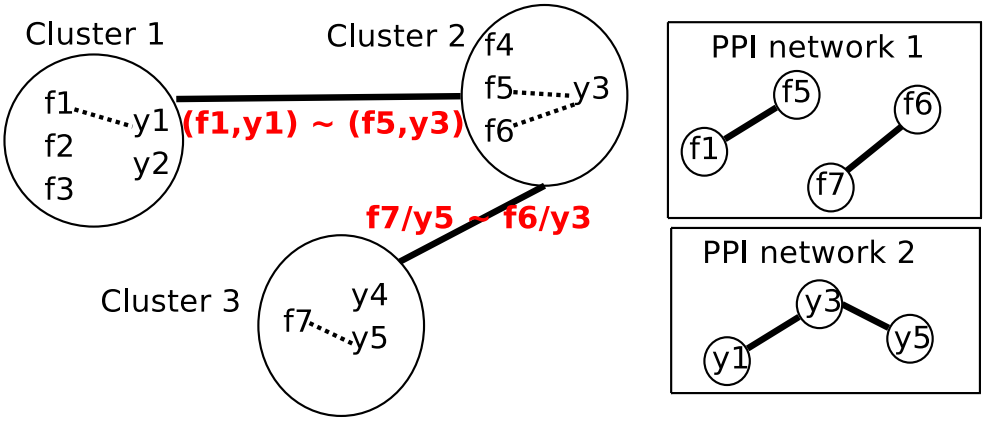


Figure 3.1: Inparanoid cluster network. Two clusters are connected if there exist at least one pair of proteins in one cluster, and one pair of proteins in the other cluster, which may produce a conserved interaction.

and y_1 from the second organism, which interact with f_5 and y_3 respectively, both in c_2. The reason why we introduce this graph of clusters is that it allows to decompose the choice of a global matching P into local matchings within each cluster, the dependency between the local choices being described by the edges of the graph. For example, if a cluster is isolated, then the choice of the matching within this cluster has no influence over the total number of conserved interactions apart from interactions within this cluster. In other words, the local matching within an isolated cluster can be optimized independently from the others. On the other hand, if a cluster is connected to other clusters, then changing the matching within this cluster can affect the total number of interactions between proteins of different clusters, and the matchings between connected clusters must be chosen synchronously to optimize the total number of conserved interactions.

More formally, if we denote by $P_1, \ldots, P_L$ the permutation P restricted to the L clusters, then an important property is that the total number of interactions

conserved by P decomposes as:

$$J(P) = \sum_{i=1}^{L} J_1(P_i) + \sum_{i \sim j} J_2(P_i, P_j)\,, \tag{3.9}$$

where $J_1(P_i)$ denotes the number of conserved interactions within c_i, $J_2(P_i, P_j)$ denotes the number of conserved interactions between c_i and c_j, and $i \sim j$ means that c_i is connected to c_j.

While maximizing (3.9) remains a challenging optimization problem in general, it may be optimized efficiently if the graph of clusters has a particular structure, e.g., if many nodes are isolated or if it contains no loop. For example, Figure 3.2(a) shows the graph of clusters for the problem of fly/yeast protein alignment investigated by Bandyopadhyay et al. [2006]. Interestingly, this graph has no loop. In this case, we can maximize (3.9) by a particular Message Passing (MP) algorithm [Jordan, 2001]. The idea of the MP algorithm is similar to the Viterbi algorithm [Viterbi, 1973] widely used to optimize functions over linear graphs, such as finding the most likely set of hidden states in a hidden Markov model [Durbin et al., 1998]. Here we describe how to apply MP on a graph without loop to optimize (3.9). First, we note that each of the permutations involving proteins within a connected component of the graph can be optimized independently from each other, so we just consider a single connected component without loop, i.e., a tree $\mathcal{T}$ of clusters. We choose a vertex of $\mathcal{T}$ that we call root, which allows to define the directions up (towards the root) or down (away from the root) when moving on edges of the graph. Each cluster c_i except the root has a unique parent cluster, namely, the connected cluster in the direction of the root. The clusters connected to a cluster c which are not its parent are called its children and are denoted $ch(c)$. To each node c of $\mathcal{T}$, we associate a vector $u_c \in \mathbb{R}^{\mathcal{P}_c}$, where $\mathcal{P}_c$ is the set of possible local matchings within c, i.e., the set of possible P_c's. The MP algorithm to solve (3.9) is then a recursive algorithm, which starts from the leaves up to the root in a first phase (the "forward" step) to find the optimal value of the functional, and then downwards from the root

to leaves (the "backward" step) to find the solution which achieves the optimal value. The forward step at node c solves, for any $P_c \in \mathcal{P}_c$:

$$u_c(P_c) = J_1(P_c) + \sum_{c' \in ch(c)} \max_{P_{c'} \in \mathcal{P}_{c'}} \left[u_{c'}(P_{c'}) + J_2(P_c, P_{c'}) \right] . \tag{3.10}$$

At the end of the forward step, the maximum value of the vector u at the root is equal to the maximal value of $J(P)$, and the local permutation which achieves this maximum is the optimal local permutation. In the backward step, the optimal local matching of the children of a cluster are obtained by recovering the local permutations $P_{c'}$ which achieved the optimal value in (3.10) for the optimal permutation of the parent cluster.

We note that it is also possible to use the MP algorithm on graphs that are not trees, but which have a small tree-width value [Jordan, 2001]. Roughly speaking it means that the graph of clusters is not a tree, we may transform it into a tree by grouping together clusters. If the size of these cluster groups is not very large, then the exact optimization may still be feasible.

3.4 Data

In order to compare the performance of the different graph matching methods, we performed several experiments aiming at aligning the PPI networks of the yeast *S. cerevisiae* and of the fly *D. melanogaster*, as already investigated by Bandyopadhyay et al. [2006] and Singh et al. [2008]. We downloaded all necessary data from the supplementary materials of Bandyopadhyay et al. [2006][2]. The yeast PPI network contains 4,389 proteins and 14,319 pairwise interactions, while the fly network contains 7,038 proteins and 20,720 interactions. In addition we also retrieved the set of Inparanoid clusters used by Bandyopadhyay et al. [2006], consisting in 2,244 cluster covering 2,834 yeast proteins and 3,881 fly proteins. The majority of these clusters (1,552) contains only two proteins (one from fly, one from yeast), while the remaining 692

[2] `http://www.cellcircuits.org/Bandyopadhyay2006http://www.cellcircuits.org/Bandyopadhyay2006`

cluster contain at least two proteins from the same species and one from the other species. Those 692 clusters are called ambiguous in [Bandyopadhyay et al., 2006], since they do not allow to associate a single protein from the fly to a single protein from the yeast as functional orthologs.

3.5 Results

We wish to investigate two different questions: (i) compare the ability of the different methods to find alignment with many conserved interactions, and (ii) assess whether conserving more interactions really helps in retrieving more functional orthologs. While the first question can be answered without ambiguity by counting the number of conserved interactions found by the different methods in different settings, the second one, as we will see, remains difficult to answer due to the lack of large-scale and curated ground truth.

We performed three sets of experiments, in order to compare the different methods in different settings and to test different formulations of the GNA problem. In the first set of experiments, we reproduce the problem studied by Bandyopadhyay et al. [2006], where the goal is to disambiguate functional orthologs within Inparanoid clusters using PPI information. This is a particular instance of the constrained GNA problem which turns out to be amenable to exact optimization by the MP method. In the second set of experiments, we generalize the benchmark problem of Bandyopadhyay et al. [2006] by adding second-order interactions between proteins in order to account for possible noise in the interaction data or protein duplications. In that case we are again confronted with a constrained GNA problem, but the increased number of interactions makes its exact minimization intractable and only approximate methods for constrained GNA can be applied. Finally, in a third set of experiments, we discard the knowledge of Inparanoid clusters and directly search a global alignment which balances the similarity between aligned proteins and the number of conserved interactions. This is then an instance of the balanced GNA problem. In all cases, we assess the number of conserved interactions captured by the different methods, as an indicator of how well they solve the GNA problem. Furthermore,

since the final objective of PPI network alignment is to match functional orthologs, we assess for each method how many matched pairs are present in the HomoloGene database, a set of curated functional orthologous pairs based on the comparison of the protein as well as the DNA sequence which we consider here as a "gold standard" for disambiguation purpose.

3.5.1 Disambiguation of functional orthologs within Inparanoid clusters

The goal of this experiment is to use PPI GNA to select functional orthologs between the yeast and the fly for proteins with several homologs. More precisely, all proteins sequences are first clustered into groups by the Inparanoid algorithm [Brein et al., 2005], and only proteins from the same cluster can be considered as protein functional orthologs. Then each GNA algorithm tries to find an association of protein functional orthologs which maximizes the total number of conserved interactions. In other words, we try to solve the constrained GNA (3.2), where the constraints are provided by the Inparanoid clusters. A priori, the most natural definition of "conserved interaction" for the alignment $(f_1 - y_1)$ and $(f_2 - y_2)$ (where f_1 and f_2 are fly's proteins, and y_1 and y_2 are yeast's proteins) is the following:

1. f_1 interacts with f_2, and y_1 interacts with y_2 in their respective PPI networks.

However, this strict notion of conserved interaction leads to a very small number of potentially conserved interactions. To have more potential interactions, Bandyopadhyay et al. [2006] generalized this definition by adding the following two cases, which additionally allow to account for possible duplication or fusion events in the two proteomes:

2. f_1 interacts with f_2 in the fly PPI network, and y_1 has a common neighbor with y_2 in the yeast PPI networks;

3. f_1 has a common neighbor with f_2 in the fly PPI network, and y_1 interacts with y_2 in the yeast PPI networks.

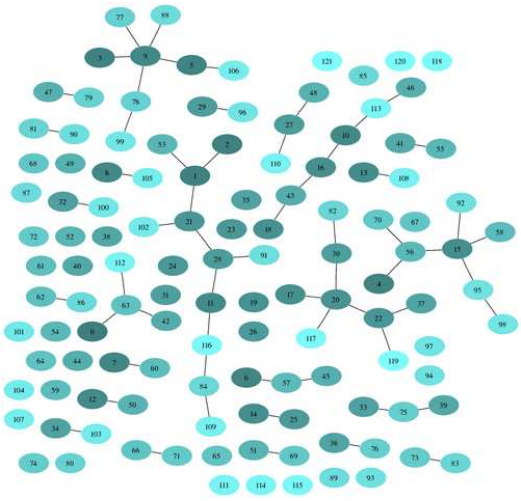
Table 3.1: Performance of the different methods for constrained GNA on the benchmark of Bandyopadhyay et al. [2006]. Each algorithm is evaluated by the number of conserved interactions, number of recovered HomoloGene pairs and the running time. The number of recovered HomoloGene pairs is counted only in 121 ambiguous Inparanoid clusters where PPI data may be used.

Algorithm	MP	MRF	IsoRank	GA	PATH
Number of conserved interactions	**238**	233	228	**238**	**238**
Number of HomoloGene pairs (121 cl.)	41	36	39	41	41
Timing(sec)	1-2	10	1-2	1-2	80-100

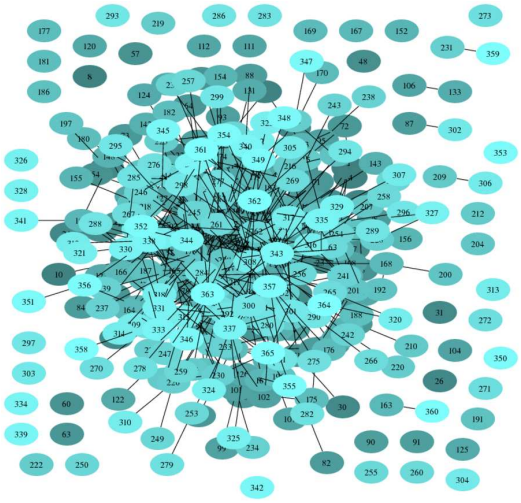
To be able to compare the results of different algorithms, we use this exact definition of conserved interactions (cases 1-3). Figure 3.2(a) presents the network of Inparanoid clusters (as explained in Figure 3.1) used in [Bandyopadhyay et al., 2006], where only non-isolated ambiguous clusters are shown. As can be easily seen, this network which contains 121 ambiguous clusters has no loop, which implies that we can use the MP method to find the optimal alignment with the largest number of conserved interactions. Although we know how to solve the problem exactly in this case with the MP method, it is instructive to compare also the results of the different approximate algorithms for constrained GNA, namely, MRF and the constrained versions of IsoRank, GA and PATH. To construct the alignment made by the MRF method [Bandyopadhyay et al., 2006], we downloaded the result file[3] with probabilities for all possible protein association, and we extracted the one-to-one alignment by taking the most probable pairs. The results of the PATH, GA and IsoRank algorithms were obtained with the GraphM package [Zaslavskiy et al., 2008a].

Table 3.1 presents the results of all algorithms on this benchmark, in terms of conserved interactions, number of HomoloGene pairs, and running time. We know that the MP algorithm produces the maximal possible value (238 in this case), and an interesting observation is that the GA and the PATH algorithms reach this maximum, while the MRF (233) and the IsoRank (228) algorithms do not. All methods are comparable in terms of CPU time, except for MRF which is one order of magnitude slower on this dataset. Although the differences in number are slight, with only 2%

[3]http://www.cellcircuits.org/Bandyopadhyay2006/data/Bandyopadhyay_results.xls



(a)



(b)

Figure 3.2: Inparanoid cluster networks. (a) The case of the benchmark data used in [Bandyopadhyay et al., 2006]. (b) The case of generalized interactions (1-4), see text.

more conserved interactions for MP/GA/PATH than for MRF, and 4% more than for IsoRank, this nevertheless confirms that even on this relatively easy optimization problem neither MRF nor IsoRank finds the optimal solution, which can be found by other methods at no additional computational cost.

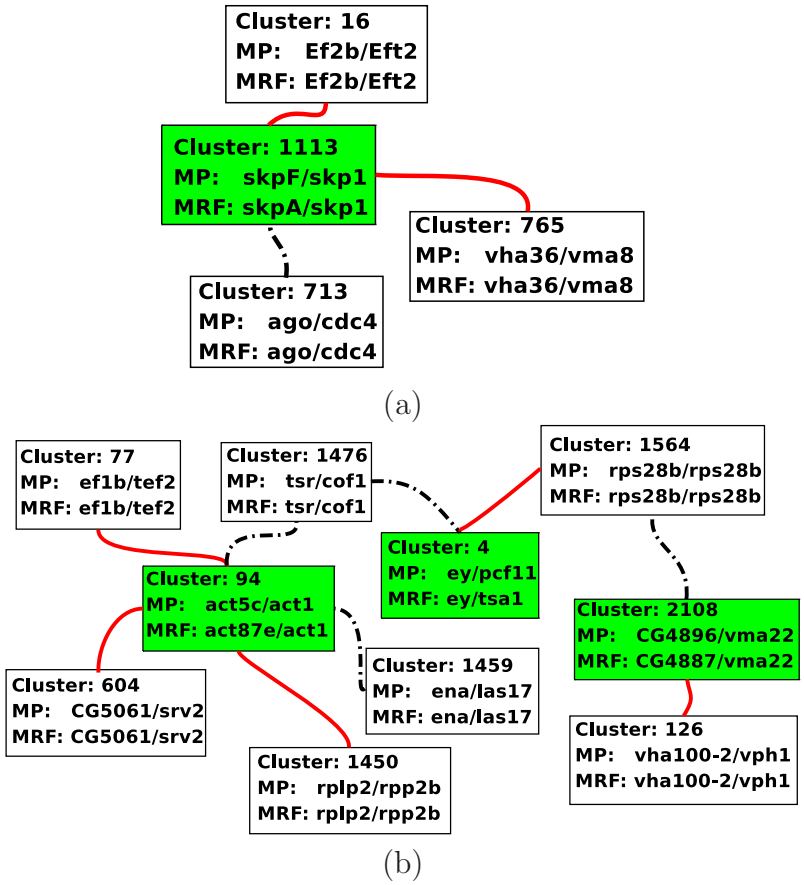


Figure 3.3: Illustration of difference between MRF and MP alignment. Each box represents an Inparanoid cluster, white unfilled boxes represent clusters where MP and MRF assignments are the same. Red solid lines represent interactions conserved by MP alignment and not by MRF, black dotted lines represent interactions conserved by MRF and not by MP.

Figures 3.3(a) and 3.3(b) show some examples where the MRF assignment and the assignment made by the MP, PATH and GA algorithms are different, and illustrate how these differences influence the total number of conserved interactions. For

instance, in the Inparanoid cluster 1113, the MRF algorithm associate the fly protein skpA to the yeast protein skp1, while the MP algorithm prefers the assignment skpF to skp1. In the later case we lose one conserved interaction with pair ago-cdc4, but we gain two new conserved interactions with (vha36,vm28) and (ef2b,eft2). In another example, shown in Figure 3.3(b), the MP algorithm proposes a different association for the yeast protein act1 in the 94-th Inparanoid cluster. This assignment results in two lost and three gained conserved interactions. From a biological point of view, the assignment of the fly protein act87e to act1 proposed by the MRF algorithm seems to be worse that the assignment (act5c,act1) proposed by the MP algorithm. Indeed, although proteins act5c and act87e are very similar (being both from the actine family), it is known that act1 and act5c participate together to the INO80 protein complex (which exhibits chromatin remodeling activity and 3' to 5' DNA helicase activity), while act87e does not.

In order to assess more systematically and quantitatively whether differences in the number of conserved interactions lead to significant differences in number of correctly assigned functional orthologous pairs, we counted how many pairs in each alignment is reported as functional orthologous in the HomoloGene database, considered here as a "gold standard". As shown in Table 3.1, the number of HomoloGene pairs in each alignment also differs between the different methods, ranging from 36 for MRF to 39 for IsoRank and 41 for MP/GA/PATH. Interestingly, we observe that the method MP, GA and PATH, which retrieve the largest number of conserved interaction, also result in the largest number HomoloGene pairs (41), which represents a relative increase of 13% compared to MRF (36), and of 5% compared to IsoRank. To illustrate the differences between the methods, Table 3.2 lists the HomoloGene pairs found by MRF and not MP/GA/PATH, and vice versa. Interestingly, a new method for PPI network alignment was published recently [Yosef et al., 2008], which detects 37 HomoloGene orthologs on the same set of proteins. This puts its between MRF and IsoRank according to this criterion.

The validity of taking HomoloGene as a "gold standard" for assessing the number of correctly assigned homologous pairs remains, however, subject to discussion. Indeed, although HomoloGene clusters are defined using a variety of evidences, they are

Table 3.2: HomoloGene orthologs found by the MP method and not by MRF and vice versa.

MP		MRF
(TfIIA-S, TOA2)	(RPL23, RPL23A)	(Pros35, PRE5)
(CG13890, ECI1)	(Gapdh1, TDH1)	(Rab11, Ypt31)
(TfIIS, DST1)	(Rpt4, Rpt4)	(Rps26, Rps26A)
(Ef1gamma, TEF4)	(act5c, act1)	(CG6523, YDR098C)
(Glut1, YBR241C)	(Sir2, hst1)	(CG8690, YBR299W)

mainly driven by sequence similarity. To illustrate this, we assessed the performance of a simple alignment method which matches pairs within an ambiguous cluster by maximizing the total sequence similarity over matched pairs. This method does not use any PPI information for the matching. The resulting alignment has only 184 conserved interaction, which is not surprisingly much worse than all methods which take PPI into account. However, the resulting matched pairs contain 43 HomoloGene pairs, which is more than all methods taking into account PPI. This shows that the number of HomoloGene pairs as an indicator should be taken with caution, since it favors methods which focus on matching proteins based on sequence similarity only.

3.5.2 Disambiguation of Inparanoid clusters with second-order interactions

The idea of Bandyopadhyay et al. [2006] to expand the natural notion of conserved interaction (case 1) to cases 2 and 3, aims to take into account second-order interactions, that is, when two proteins do not interact directly to each other have a common neighbor. Another natural generalization of the notion of conserved interaction is then the following case:

4. f_1 has a common neighbor with f_2, and y_1 has a common neighbor with y_2, in their respective PPI networks.

Adding interactions according to this rule makes the problem computationally more difficult, since ambiguous clusters become more connected. Indeed, while we were able to solve the original problem exactly with the MP algorithm, the network of

Table 3.3: Performance of the different methods for constrained GNA on the benchmark of Bandyopadhyay et al. [2006] with second-order interactions added. The number of recovered HomoloGene pairs is counting on the 121 Inparanoid clusters from the previous section as well as on the new 602 ambiguous Inparanoid clusters have second-order interaction with other Inparanoid clusters

Algorithm	MRF	IsoRank	GA	PATH
Number of conserved interactions	1,112	1,101	1,140	1,143
Number of HomoloGene pairs (121 cl.)	39	38	41	40
Number of HomoloGene pairs (602 cl.)	172	167	172	166
Timing(sec)	623	31	372	1,542

Inparanoid clusters when cases 1-4 are included takes the form presented in Figure 3.2(b). Contrary to the previous network (cases 1-3 in Figure 3.2(a)), the new network has loops and is not amenable to exact optimization with the MP procedure. Only approximate algorithms can be applied in this case.

In order to compare all methods (except MP) in this new setting, we re-implemented the MRF algorithm with the new data. The estimated values of the model parameters (see details in [Bandyopadhyay et al., 2006]) are ($\alpha = 0.51, \beta = -6.87$). We used the same training and test data as those used used in [Bandyopadhyay et al., 2006] to estimate them. Then we estimated the probabilities of being protein orthologs for potential pairs of proteins by Gibbs sampling, and obtained a one-to-one alignment based on the most probable associations.

Table 3.3 shows the results obtained by the different graph matching algorithms. Although we do not know the maximum number of interactions that can be conserved in this case, we observe again that PATH and GA find solutions with $3-4\%$ more interactions conserved than MRF and IsoRank. There is no clear difference in the number of HomoloGene pairs between the different methods, and the addition of second-order interactions has no obvious effects on this indicator neither: it leads to a gain of 3 pairs for MRF, but to a loss of one pair for IsoRank and PATH, and to no change for GA.

3.5.3 Global PPI network alignment by balancing sequence and interaction conservation

In this last series of experiments, we consider the problem proposed by Singh et al. [2008], for which IsoRank reflects the state-of-the-art: find a global PPI alignment by balancing the sequence similarity in matched pairs with the total number of conserved interactions, allowing in particular matches between proteins in different Inparanoid clusters if they allow an increased number of conserved interactions. For this application we can only compare the three methods for balanced GNA, namely, IsoRank, GA and PATH. The trade-off between matching proteins with similar sequences and matching with a lot of conserved interactions is controlled by the parameter λ in (3.4) and (3.7). The greater λ, the more attention we pay to the sequence similarity and the less to the number of conserved interactions. For each method, by varying λ, we therefore obtain a family of alignments with different compromise found between the number of conserved interactions $J(P)$ (3.4) and the summary sequence similarity score $S(P)$ (3.4).

Figure 3.4 shows the different trade-offs which are found by the different methods. For a given level of average sequence similarity, we wish to have the largest possible number of conserved pairs. We observe that over all the range of average sequence similarity, the GA algorithms clearly outperforms PATH, which itself outperforms IsoRank. For example, for the trade-off parameter choice advocated by Singh et al. [2008] for IsoRank ($\lambda = 0.6$), IsoRank finds an alignment with 566 conserved interactions, corresponding to an average sequence similarity score in the matched pairs of 15.26. At this level of average sequence similarity, PATH and GA find alignments with respectively 678 and $1,006$ interactions, which corresponds to relative improvements of respectively 20% and 78%.

Again, there is still only limited objective evidence that optimizing the number of conserved interactions leads to better matching in terms of functional orthology detection. As an attempt to test this fact, we first counted, for each alignment, the number of HomoloGene pairs in the alignment. However, we observed that, for each method, this number increases monotonically when more weight is given to

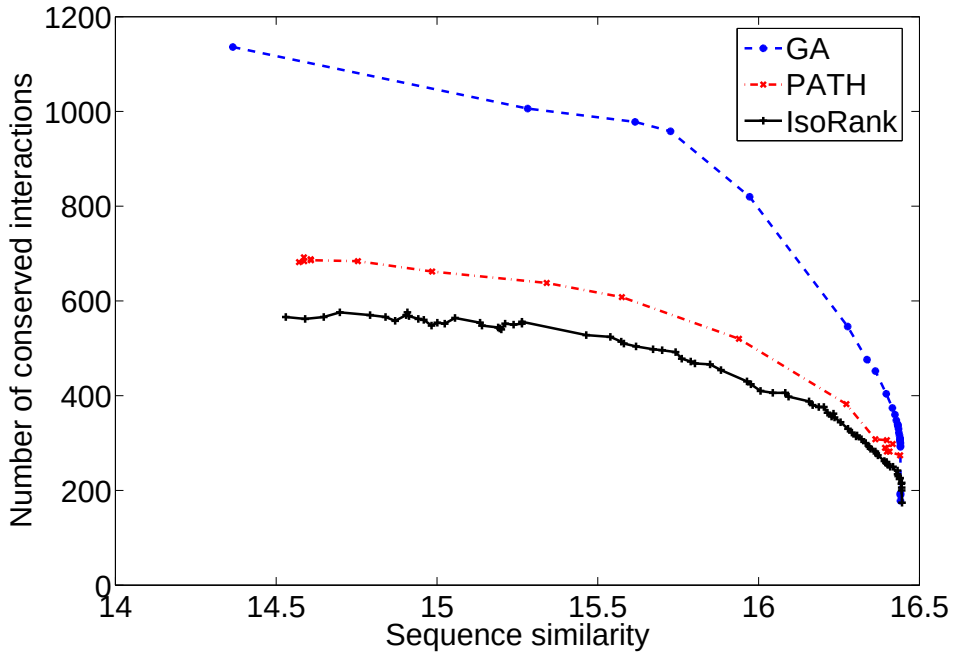


Figure 3.4: Algorithm performance comparison. Number of conserved interaction $J(P)$ versus sequence similarity $S(P)$.

sequence similarity as opposed to interaction conservation. This again highlights the limitation of this criterion, which is optimized by construction when sequences are optimally matched in terms of similarity. We then attempted to compare the different alignments in terms of mean similarity between gene ontology (GO) annotations of matched pairs. In order to compare GO annotations of two proteins we tested the method presented by Singh et al. [2008] to compute the functional coherence of a pair. However, we were not able to observe any clear difference between the methods, or between the different parameter choice for each individual method. The maximum mean functional coherence over the choice of the trade-off parameter is respectively 0.519, 0.509 and 0.522 for IsoRank, GA and PATH. However the fluctuations of this score when the parameters change are so large that these maximum values are not significantly different. This is due to the fact that the number of annotated proteins remains limited, and that they are rarely annotated with such precision that it is possible to clearly differentiate true functional orthologs from spurious ones [Bandyopadhyay et al., 2006]. For example, when we estimate the functional score of a given alignment, there is rarely more than $15 - 20\%$ of pairs with GO annotations.

3.6 Discussion

We presented two general formulations for the GNA problem. The constrained GNA formulation corresponds to a situation where we have a strong *a priori* about which pairs can be matched. In the balanced GNA problem, we replace the binary constraints on which pairs are allowed by a more global objective function which balances the matching of similar proteins with the conservation of interactions, with a parameter to smoothly control the trade-off between these two contradictory goals. While MRF and IsoRank are popular methods for these two formulations, we proposed in this chapter new methods which lead to significantly better alignments, when we assess the quality of an alignment in terms how many conserved interactions are retrieved. In particular, the MP method, when it is applicable, finds the optimal solution of a constrained GNA problem, and the GA method provides consistently good results in both cases. The question of which formulation is the best for a given

application and dataset, between the constrained and balanced GNA, remains largely open and worth further systematic investigations. Regarding the relative performance of the different methods in terms of how many conserved interactions they find, we observed that the MP/GA/PATH methods outperform MRF and IsoRank in both situations. This is not so surprising given that, once the problem is explicitly stated as a graph matching problems, it makes sense to use methods borrowing ideas and techniques from state-of-the-art graph matching approaches. The impressive performance of GA compared to PATH in the balanced GNA experiment (Figure 3.4) is more surprising, given the good performance of PATH on a number of other benchmarks [Zaslavskiy et al., 2008c]. We believe this weakness of PATH is due to the large difference in the number of nodes between the two networks. Indeed, the resulting large number of dummy nodes that must be added generate singularities in the convex relaxation in the PATH algorithm.

The GNA problems we studied have several extensions. First, it may be interesting to consider alignment of weighted PPI networks with weights representing, for instance, experimental evidence of interaction existence. Interestingly, the PATH, GA and IsoRank algorithm can be applied directly to a weighted network, by just replacing the binary graph adjacency matrix by a real-valued matrix. Another relevant extension is the alignment of multiple PPI networks, corresponding to more than two species, via pairwise comparisons as it was presented by Singh et al. [2008]. Finally, it may be relevant in some cases to match one protein of one species with several proteins of the other species, to account for possible duplications or fusion events. An interesting property of the PATH algorithm is the fact that estimate a permutation matrix by first solving a relaxed problem. The solution of the relaxed problem is a doubly stochastic matrix whose entries can be interpreted as probabilities for proteins to be functional orthologs [Zaslavskiy et al., 2008c] . Therefore, in order to allow many-to-many assignments of proteins, we could use the solution of the convex relaxation.

Finally, although progresses in graph alignment algorithms can be monitored by objective quantitative measures such as the number of conserved interactions, their biological relevance remains difficult to assess. In particular, for the detection of

functional orthologs, it is apparent that current GO annotations or curated databases of functional orthologs are either biased by construction (e.g., HomoloGene), or not precise enough and too scarce for systematic evaluation (e.g., GO annotations). We believe we are reaching a point where more experimental validations are needed. On the other hand, there are many other possible applications for efficient graph matching algorithms scaling to large biological networks, such as phylogenetic comparison of sets of networks, detection of new conserved pathways, or curation of PPI data. We expect the methods proposed in this chapter to have a direct impact in these applications.

Chapter 4

Many-to-Many graph matching

Preface

Graphs provide a very efficient tool for object representation in various machine learning applications. Once graph-based representations are constructed, the important question of graph comparisons arises. This problem is often formulated as a graph matching problem where one seeks a mapping between vertices of two graphs which optimally aligns their structure. In the classical formulation of graph matching, only one-to-one correspondences are considered, which is not always appropriate, and in many applications it is more interesting to consider many-to-many correspondences between graph vertices: clusters of vertices in one graphs are matched to clusters of vertices in the other graph. In this chapter, we reformulate the many-to-many graph matching problem as a discrete optimization problem and we propose an approximate algorithm based on a continuous relaxation. We compare our method with other existing methods on several benchmark datasets.

4.1 Introduction

The necessity to process data with complex structures has triggered the wide use of graph-based representation techniques in various applications domains. Graphs

provide a flexible and efficient tools for data representation in computer vision (segmentation, contour and shock graphs), computational biology (biological networks), or chemoinformatics (representation of chemical compounds), to name just a few. A fundamental question when data are represented as graphs is to be able to *compare* graphs. In particular, it is important in many applications to be able to assess quantitatively the similarity between graphs (e.g., for applications in supervised or unsupervised classification), and to detect similar parts between graphs (e.g., for identification of interesting patterns in data).

Graph matching is one approach to perform these tasks. In graph matching, one tries to "align" two graphs by matching their vertices in such a way that most edges are conserved across matched vertices. Graph matching is useful both to assess the similarity between graphs (e.g., by checking how much the graphs differ after alignment), and to capture similar parts between graphs (e.g., by extracting connected sets of matched vertices).

Classically, only one-to-one mappings are considered, that is, each vertex of the first graph can be matched to only one vertex of the second graph, and vice-versa. This problem can be formulated as a discrete optimization problem, where one wishes to find a one-to-one matching which maximizes the number of conserved edges between matched pair. This problem is NP-hard for general graphs, and remains impossible to solve in practice for graphs with more than 30 vertices or so. Therefore much effort has been devoted to the development of approximate methods which are able to find a "good" solution in a reasonable time. These methods can roughly be divided into two large classes. The first group consists of various local optimization algorithms on the set of permutation matrices, including A^*-Beam-search [Neuhaus et al., 2006], genetic algorithms, GRASP [Pardalos et al., 1995] in different modifications. The second group consists in solving a continuous relaxation of the discrete optimization problem, such as the L_1 relaxation [Almohamad and Duffuaa, 1993], the Path algorithm [Zaslavskiy et al., 2008c], and various spectral relaxations [Umeyama, 1988; Caelli and Kosinov, 2004; Carcassoni and Hancock, 2003; Cour et al., 2006].

In practice, we are sometimes confronted with situations where the notion of one-to-one mapping is too restrictive, and where we would like to allow the possibility to

match groups of vertices of the first graph to groups of vertices of the second graph. We call such a mapping *many-to-many*. For instance, in image processing, the same parts of the same object may be represented by different numbers of vertices depending on the noise in the image or on the choice of object view, and it could be relevant to match together groups of vertices that represent the same part. From an algorithmic point of view, this problem has been much less investigated than the one-to-one matching problem. Some one-to-one matching methods based on local optimization over the set of permutation matrices have been extended to many-to-many matching, e.g., by considering the possibility to merge vertices and edges in the course of optimization [Berretti et al., 2004; Ambauen et al., 2003]. Spectral methods have also been extended to deal with many-to-many matching by combining the idea of spectral decomposition of graph adjacency matrices with clustering methods [Keselman et al., 2003; Caelli and Kosinov, 2004]. However, while the spectral approach for one-to-one matching can be interpreted as a particular continuous relaxation of the discrete optimization problem [Umeyama, 1988], this interpretation is lost in the extension to many-to-many matching. In fact, we are not aware of a proper formulation of the many-to-many graph matching problem as an optimization problem solved by relaxation techniques.

Our main contribution is to propose such a formulation of many-to-many graph matching problem as a discrete optimization problem which generalizes (4.1), and to present an approximate methods based on a continuous relaxations of the problem. The relaxed problem is not convex, and we solve it approximately with a conditional gradient method. We also study different ways to map back the continuous solution of the relaxed problem into a many-to-many matching. We present experimental evidence, both on simulated and real data, the this formulation provides a significant advantage over other one-to-one or many-to-many matching approaches.

4.2 Many-to-many graph matching as an optimization problem

In this section we derive a formulation of the many-to-many graph matching problem as a discrete optimization problem. We start by recalling the classical expression of the one-to-one matching problem as an optimization problem. We then show how the one-to-one formulation may be extended to the case of one-to-many matchings. Finally we describe how we can define many-to-many matchings via two many-to-one mappings.

One-to-one graph matching Let G and H be two graphs with N vertices. We also denote by G and H their respective adjacency matrices. A one-to-one matching between G and H can formally be represented by a $N \times N$ permutation matrix P, where $P_{ij} = 1$ if the i-th vertex of graph G is matched to the j-th vertex of graph H, and $P_{ij} = 0$ otherwise. Denoting by $\| \dots \|_F$ the Frobenius norm of matrices, defined as $\|A\|_F^2 = \mathrm{tr} A^T A = (\sum_i \sum_j A_{ij}^2)$, we note that $\|G - PHP^T\|_F^2$ is twice the number of edges which are not conserved in the matching defined by the permutation P. The one-to-one graph matching problem is therefore classically expressed as the following discrete optimization problem:

$$\begin{aligned} &\min_{P \in \mathcal{P}} ||G - PHP^T||_F^2 \\ &\text{subject to } P \in \mathcal{P}_{Odo} = \left\{ P \in \{0,1\}^{N \times N} : P 1_N = 1_N, P^T 1_N = 1_N \right\} . \end{aligned} \tag{4.1}$$

We note that $\mathcal{P}_{oto}$ simply represents the set of permutation matrices.

From one-to-one to one-to-many Suppose now that G has less vertices than H, and that our goal is to find a matching that associate each vertex of G with one or more vertices of H in such a way that all vertices of H are matched to a vertex of G. We call such a matching *one-to-many* (or many-to-one). The problem of finding an optimal one-to-many can be formulated as minimizing the same criterion as (4.1)

but modifying the optimization set as follows:

$$\mathcal{P}_{otm} = \{P \in \{0,1\}^{N_G \times N_H} : P1_{N_H} \leq k_{\max} 1_{N_G},\ P1_{N_H} \geq 1_{N_G},\ P^T 1_{N_G} = 1_{N_H}\}\ , \tag{4.2}$$

where N_G denotes size of graph G, N_H denotes size of graph H, and $k_{\max}$ denotes an optional upper bound on the number of vertices that can be matched to a single vertex. In the context of one-to-many graph matching when g is matched to $h_1, \ldots, h_n$, we can see it as merging $\mathbf{h} \leftarrow h_1 + h_2 + \cdots + h_n$ with further matching g to $\mathbf{h}$, where meta-vertex $\mathbf{h}$ inherits all neighborhood connections (edges) from all $h_1, \ldots, h_n$ i.e. $w(\mathbf{h}, h') = \sum_{i=1}^{n} w(h_i, h')$.

The majority of existing continuous relaxation techniques may be adopted for one-to-many matching. For example, Cour et al. [2006] describes how spectral relaxation methods may be used in the case of one-to-many matching, other techniques like convex relaxation [Zaslavskiy et al., 2008c] may be used as well since $\mathcal{P}_{otm}$ is a convex set.

From one-to-many to many-to-many Now to match two graphs G and H under many-to-many constraints we proceed as if we matched these two graphs to a virtual graph S under many-to-one constraints, minimizing the difference between transformed graph G and transformed graph H. The idea of many-to-many matching as a double one-to-many matching is illustrated in Figure 4.1. Graph S represents

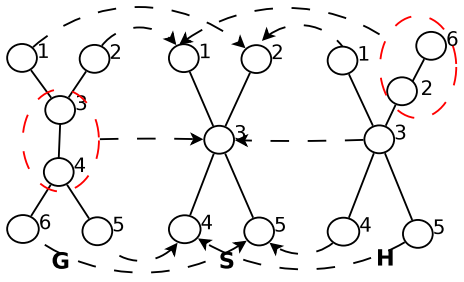


Figure 4.1: Many-to-Many matching between G and H via many-to-one matching of both graphs to a virtual graph S

the graph of matched vertex clusters. Each vertex of S corresponds to a group of

vertices of G and a group of vertices of H matched to each other. Let P_1 denote a many-to-one matching $G \to S$, and P_2 a many-to one matching $H \to S$, then the many-to-many graph matching problem may be formulated as an optimization problem where we seek S, P_1 and P_2 which minimize the difference between S and $P_1 G P_1^T$ and between S and $P_2 H P_2^T$. The intermediate graph S may be squeezed out by considering directly the difference between $P_1 G P_1^T$ and $P_2 H P_2^T$. We end up with the following objective function for the many-to-many GM problem:

$$F(P_1, P_2) = ||P_1 G_1^T P_1^T - P_2 H P_2^T||_F^2, \tag{4.3}$$

where $P_1 \in \mathcal{P}_{mto}$ and $P_2 \in \mathcal{P}_{mto}$ denote two many-to-one mappings. The objective function (4.3) is quite similar to the objective function in the one-to-one case. In (4.1) we seek a permutation which makes the second graph H as similar as possible to G. And here in the many-to-many case we seek combinations of merging and permutations which makes G and H similar to each other. The only difference is that in the many-to-many case we add the merging operation.

There are two slightly different ways of defining the optimization set. We can fix in advance the number of matching clusters L which corresponds to the size of S, then the optimization set will be $P_1 \in \mathcal{P}_{mto}(L, N_G)$ and $P_2 \in \mathcal{P}_{mto}(L, N_H)$. An alternative way is to remove the constraint $P 1_{N_H} \geq 1_{N_G}$ from the definition of $\mathcal{P}_{mto}$, in this case the method estimates itself the number of matching clusters (number of rows with non-zero sum).

Finally, the many-to-many graph matching problem is formulated as follows

$$\begin{aligned} &\min_{P_1, P_2} ||P_1 G_1^T P_1^T - P_2 H P_2^T||_F^2 \quad \text{subject to} \\ &P_1 \in \{0,1\}^{N_K \times N_G},\ P_1 1_{N_G} \leq k_{max} 1_{N_K},\ P_1' 1_{N_K} = 1_{N_G}, \\ &P_2 \in \{0,1\}^{N_K \times N_H},\ P_2 1_{N_H} \leq k_{max} 1_{N_K},\ P_2' 1_{N_K} = 1_{N_H}\ , \end{aligned} \tag{4.4}$$

where $N_K = \min(N_G, N_H)$ represent the maximal number of matching clusters.

This formulation is valid for all kinds of graphs, graphs may be directed (with asymmetric adjacency matrices), have real-valued edge weights and self-loops. We

also describe in Section 4.3 how this formulation may be modified to include information on vertex labels.

4.3 Continuous relaxations of the many-to-many GM problem

The many-to-many graph matching problem (4.4) is a hard discrete optimization problem and therefore an approximate method is needed. Here we propose an algorithm based on a continuous relaxation of (4.4)

4.3.1 Method 1: Gradient descent

In this method we replace the binary constraints $P_1 \in \{0,1\}^{N_K \times N_G}$, $P_2 \in \{0,1\}^{N_K \times N_H}$ by continuous constraints $P_1 \in [0,1]^{N_K \times N_G}$, $P_2 \in [0,1]^{N_K \times N_H}$. Then, we run the following optimization algorithm (which is a version of the Frank-Wolfe or the conditional gradient method [Bertsekas, 1999])

- Input: initial values P_1^0 and P_2^0
- while NOT(stop_criterion)
  1. compute $\nabla F(P_1, P_2)$
  2. find the minimum $\nabla F(P_1, P_2)P$ w.r.t. P; this problem may be seen as a version of a linear semi-assigned problem and reduced to the classical linear assignment problem by adding dummy nodes. Finally we have to solve a linear assignment problem for $k_{max}(N_G + N_H) \times N_H$ matrix which can be done very efficiently by the Hungarian algorithm [Kuhn, 1955].
  3. do the line search in the direction of the optimum find in Step 2. Since the objective function is a polynomial of the fourth order, there exists a closed form expression for the minima of $F(P_1, P_2)$ in any given direction.
  4. set stop_criterion to true if $|\Delta F| + ||\Delta P_1||_F + ||\Delta P_2||_F < \epsilon$

- Output P_1, P_2.

Since the objective function is not convex, it is very important to have a good initialization. In our experiments the method is working well with the "uniform" initialization, where we initialize P_1 by $\frac{1}{N_K} 1_{N_G} 1_{N_H}^T$ and P_2 by the identity matrix I. Another option is to use a convex relaxation of one-to-one matching [Zaslavskiy et al., 2008c]

Projection The last step consists in the projections of P_1 and P_2 onto $\mathcal{P}$. Here, we have several alternatives. Columns of matrices P_1 and P_2 may be used to define a similarity measure between vertices of two graphs. The more similar the corresponding columns of two vertices, the more likely these vertices are to be matched (if they are from different graphs) or merged (if they are from the same graph). Therefore a possible strategy is to run a clustering algorithm (K-means, spectral clustering,...) on column vectors of matrix P and then use resulting clustering to construct the final many-to-many graph matching.

An alternative to clustering is an incremental projection or forward selection projection. Once P_1 and P_2 are constructed we take the pair of vertices g and h having the most similar column vectors and we fix it as a matched pair. To fix a pair of vertices as matched pair, we set $P(:,g) \leftarrow P(:,h) \leftarrow e_q$. If g (or h) is already matched to another vertex h', then we set $P(:,h) \leftarrow P(:,g)$ producing a one-to-two matching $g \sim (h, h_1)$. Then we add the new constraints $P(:,g) = P(:,h) = e_q$ to the optimization problem and we adjust the minima. We repeat this operation until the moment when all vertices are matched.

In our experiments the second approach produced better results mainly due to the fact that when we just run a clustering algorithm we do not use the objective function, while when we use incremental projection we adapt column vectors of unmatched vertices according to earlier established matchings.

4.3.2 Method 2: SDP relaxation

The second method consists in relaxation of (4.4) to a quadratic SDP problem.

First, we rewrite the objective function of (4.3) in an alternative form

$$
\begin{aligned}
&||P_1 G_1^T P_1^T - P_2 H P_2^T||_F^2 = \\
&\mathrm{tr} P_1 G^T \underbrace{P_1^T P_1}_{M_1} G P_1^T + \mathrm{tr} P_2 H^T \underbrace{P_2^T P_2}_{M_2} H P_2^T - 2\mathrm{tr} P_1 G^T \underbrace{P_1^T P_2}_{M_{12}} H P_2^T = \\
&\mathrm{tr} M_1 G^T M_1 G + \mathrm{tr} M_2 H^T M_2 H - 2\mathrm{tr} M_{21} G^T M_{12} H = \\
&\mathrm{tr}\left[\underbrace{\begin{pmatrix} M_1 & M_{12} \\ M_{21} & M_2 \end{pmatrix}}_{M} \underbrace{\begin{pmatrix} G^T & 0 \\ 0 & -H^T \end{pmatrix}}_{A^T} \begin{pmatrix} M_1 & M_{12} \\ M_{21} & M_2 \end{pmatrix} \begin{pmatrix} G & 0 \\ 0 & -H \end{pmatrix} \right] = \\
&\mathrm{tr} M A^T M A = \mathrm{vec}(M)(A^T \otimes A)\mathrm{vec}(M) =: F(M)
\end{aligned}
\tag{4.5}
$$

We have to minimize a quadratic function over discrete set $\mathcal{M}$ of binary matrices having special structure. Since matrix M is a positive-semidefinite matrix, we can relax the optimization problem $\min_{M \in \mathcal{M}} F(M)$ to the minimization of a quadratic function over the convex set of positive-semidefinite matrices

$$\min_{M \succeq 0} F(M). \tag{4.6}$$

Therefore the second method consists in running of the Frank-Wolfe algorithm with an SDP solver (SeDuMe, for instance) to compute conditional gradient and further projection of the produced solution on $\mathcal{M}$.

Here again we can run a clustering algorithm using matrix M_{sdp} as a similarity matrix between vertices of two graphs, or use the incremental projection strategy fixing on each step the most probable matching and adjusting the optimum given the new constraints.

Neighbor merging Depending on the particular application, it may be interesting to favorite merging of neighbor vertices, then it may be useful to consider the following modification

$$F(P_1, P_2) = F(P_1, P_2) - \mathrm{tr} G^T P_1^T P_1 - \mathrm{tr} H^T P_2^T P_2. \tag{4.7}$$

Local similarities Like the one-to-one formulation, the many-to-many graph matching may be easily modified to include information on vertex pairwise similarities by modifying the objective function as follows

$$F_\lambda(P_1, P_2) = (1-\lambda)||P_1 G_1^T P_1^T - P_2 H P_2^T||_F^2 + \lambda \mathrm{tr} C^T P_1^T P_2 \ , \tag{4.8}$$

where matrix $C \in \mathbb{R}^{N_G \times N_H}$ encodes the matrix of local dissimilarities between graph vertices, and parameter λ controls the relative impact of information on graph vertices and information on graph structures. The new objective function is again a polynomial of the fourth order, so our algorithm may still be used directly without any additional modifications.

4.4 Related methods

There are two major groups of methods which may be used for many-to-many graph matching. The first one consists of local search algorithms generally used in the context of the graph edit distance. The second group are different variations of spectral approach. Below we present a brief description of these two groups.

Local search algorithms Examples of this kind of approach are given in [Berretti et al., 2004] and [Ambauen et al., 2003]. In the classical formulation of the graph edit distance, the set of graph edit operations consists of deletion, insertion and substitution of vertices and edges. Each operation has an associated cost, and the objective is to find a sequence of operations with the lowest total cost transforming one graph into another. In the case of many-to-many graph matching this set of operations is completed by merging (and splitting if necessary) operations.

Since the estimation of the optimal sequence is a hard combinatorial problem, usually approximate methods such as beam search [Neuhaus et al., 2006] as well as other examples of best-first, breadth-first and depth-first searches are used.

Spectral approach Caelli and Kosinov [2004] discuss how spectral matching may be used for many-to-many graph matching. Their algorithm is similar to the Umeyama

method but instead of one-to-one correspondences as it was done in [Umeyama, 1988], they search a many-to-many mapping by running a clustering algorithm. One the first step, the spectral decomposition of graph adjacency matrices is considered

$$G = V_G \Lambda_G V_G^T, \; H = V_H \Lambda_H V_H^T. \tag{4.9}$$

Rows of eigenvector matrices V_G and V_H may be interpreted as spectral coordinates of graph vertices. Then vertices having similar spectral coordinates are clustered together by a clustering algorithm where vertices regrouped in the same cluster are considered to be matched. Another example of spectral approach is given in [Keselman et al., 2003] where, roughly speaking, the adjacency matrix is replaced by the matrix of shortest path distances, and then spectral decomposition with further clustering are used.

4.5 Experiments

In this section we compare new methods proposed in this chapter with other existing techniques (beam-search and spectral approach). We test three competitive approaches: beam-search "Beam" (A*-beam search from [Neuhaus et al., 2006]), the spectral approach "Spec" [Caelli and Kosinov, 2004] and gradient descent "Grad" (Section 4.3).

4.5.1 Synthetic examples

In this section we compare three many-to-many graph matching algorithms on pairs of randomly generated graphs with similar structures. In our experiments we generated graphs according to the following procedure:

1. Generate a random graph G of size N: P(a pair of vertices is connected) $= p$.
2. Let H be a randomly permuted copy of G.
3. Random splitting of vertices in G (and in H): take a random vertex in G (and in H) and split it into two vertices. Repeat this operation M times.

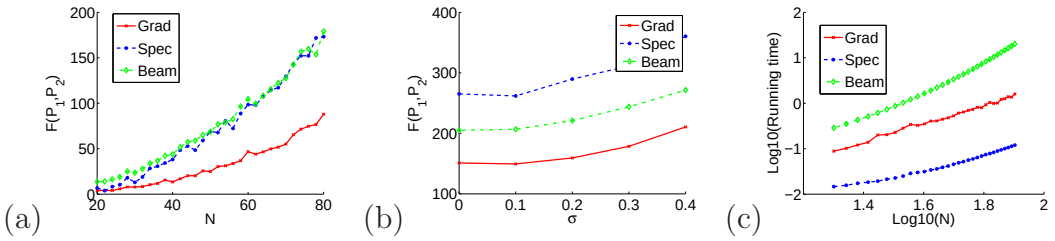


Figure 4.2: (a) $F(P_1, P_2)$ (mean value over 30 repetitions) as a function of graph size N, simulation parameters: $p = 0.1, \sigma = 0.05, M = 3$. (b) $F(P_1, P_2)$ (mean value over 30 repetitions) as a function of noise parameter σ, simulation parameters: $N = 30, p = 0.1, M = 3$. (c) Algorithm running time (mean value over 30 repetitions) as a function of N (log-log scale), other parameters are the same as in (a).

4. Noise introduction. Add/delete $\sigma \times p \times N^2$ random edges in both graphs.

As already mentioned before, our principal interest here is to understand the behavior of graph matching algorithms depending on graph size N and their ability to resist to structural noise (in practice we never have identical graphs and it is important to have a robust algorithm which is able to deal with noise in graph structures). The objective function $F(P_1, P_2)$ in (4.4) represents the quality of graph matching, so to compare different graph matching algorithms we trace $F(P_1, P_2)$ as a function of N (see Figure 4.2a), and $F(P_1, P_2)$ as a function of σ (Figure 4.2b) for three algorithms of interest. In both cases, we observe that "Grad" significantly outperforms both "Beam" and "Spec" algorithms. "Beam" was run with $B = 3$ which represent a good trade-off between quality and complexity, "Spec" was run with projection on the first two eigenvectors (variants with three and more eigenvectors were also tested, but two eigenvectors produce almost the same matching quality and it works faster) with the normalization presented in [Caelli and Kosinov, 2004]. The last Figure 4.2c shows how algorithms scale with increasing N. The "Spec" algorithm is the fastest one, but "Grad" has the same complexity order as "Spec" (corresponding curves are almost parallel lines in log-log scale, so both functions are polynomials with the same degree and different multiplication constants). "Beam" algorithm is much slower, and it also has worse complexity order.

4.5.2 Chinese characters

In this section we compare many-to-many graph matching algorithms as parts of a classification framework. There, graph matching algorithms are used to compute similarity/distance between objects of interest on the basis of their graph-based representations. As the classification problem, we chose the ETL9B dataset of Chinese characters. This dataset is well suited for our purposes, since Chinese characters may be naturally represented by graphs with variable non-trivial structures.

Figure 4.3 illustrates how "Grad" works on graphs representing Chinese characters. We see that our algorithm produces a good matching (not a perfect one however) providing a correspondence between "crucial" vertices. The characters represented in

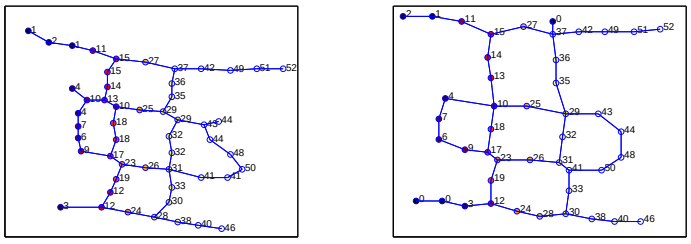


Figure 4.3: Matching of graphs representing Chinese characters made by "Grad". Vertices having the same id's are matched to each other.

Figure 4.3 are too simple for a classification problem, and the most of the classification algorithms shows good performance, for example, "Grad" produce classification error less than $0,002$. To test graph matching algorithms we chose "hard to classify" Chinese characters and we run k-nearest neighbor with graph matching algorithms used as distance measures. The dataset consists of 600 images (200 images of each class).

Table 4.1 shows classification results for three many-to-many graph matching algorithms, in addition, we also report results for SVM classifier with linear and Gaussian kernels, one-to-one matching with the Path algorithm (taken from [Zaslavskiy et al., 2008c]) and two versions of shape context method [Belongie et al., 2002]. One version (just "shape context") consists in computing polar histograms with further bipartite

graph matching. To run the "shape context+tps" method we have used the code published[1] by S.Belongie.

Graph matching algorithms are run using information on vertex coordinates (4.3), elements of matrix C are defined as $C_{ij} = e^{-(x_i-x_j)^2-(y_i-y_j)^2}$. Parameter λ in (4.3) as well as k (number of neighbors in KNN classifier) are learned via cross-validation. We see that the "Grad" algorithm shows the best performance outperforming other many-to-many graph matching algorithms as well as other competitive approaches.

Table 4.1: On the left, Chinese characters representing three class classification problem. On the right, characters classification results. (CV, STD)—mean and standard deviation of test error over cross-validation runs (five folds, 50 repetitions)

Method	CV	STD
Linear SVM	0.377	± 0.090
SVM with Gaussian kernel	0.359	± 0.076
KNN (one-to-one, Path)	0.248	± 0.075
KNN (shape context)	0.399	± 0.081
KNN (shape context+tps)	0.435	± 0.092
KNN (Grad)	0.191	± 0.063
KNN (Spec)	0.254	± 0.071
KNN (Beam)	0.283	± 0.079

4.5.3 Identification of object composite parts

While the pattern recognition framework is something interesting and important for the comparison of different graph matching algorithms, it evaluates only one aspect of graph matching algorithms, i.e., their ability to detect similar graphs. But in this case we are completely missing the aspect of how well graphs are aligned. To test the second aspect, we have performed the following series of experiments. We have chosen ten camel images from the MPEG7 dataset and we divided each images into 6 parts (see Figure 4.4: head, neck, legs, back, tail and body). This image segmentation automatically defines a partitioning of the corresponding graph: all graph vertices are labeled according to the image part which they represent. Figures 4.4 gives two

[1] `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/shape/`

illustrations of how this procedure works. A good graph matching algorithm should map vertices from corresponding image parts to each other, i.e., heads to heads, legs to legs and so on. Therefore to evaluate the matching quality of mapping, we use the following score. First, we match two graphs and then we try to predict vertex labels of one graph given the vertex labels of the second one. For instance, if vertex g_1 of the first image is matched to vertices h_1 and h_2 representing the head of the second image, then we predict that g_1 is of class "head". The better the graph matching, the smaller the prediction error and vice-versa.

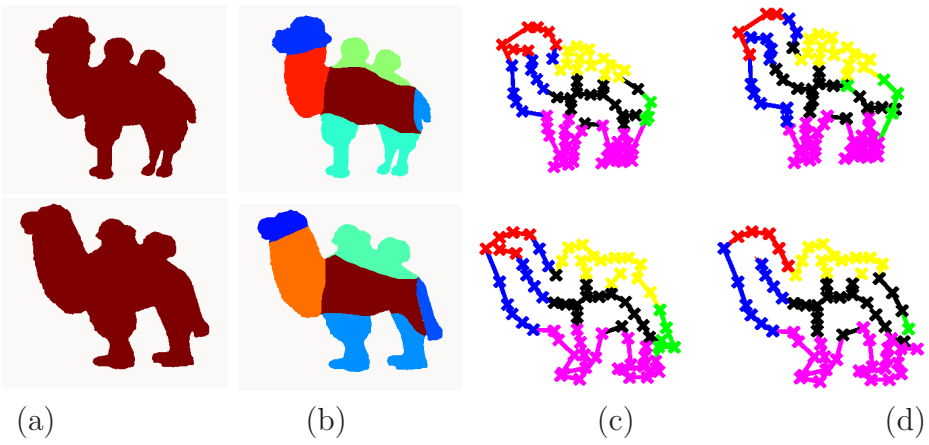


Figure 4.4: (a) Original images. (b) Manual segmentation (c) Graph-based representation with induced vertex labels (d) Prediction of vertex labels on the basis of graph matching made by "Grad".

Here we see an interesting application of graph matching algorithms. It shows how graph matching may be used for a "user defined segmentation". Usually segmentation algorithms extract image parts on the basis of different characteristics such as changing of color, narrowing of object form, etc. With our graph matching algorithm, we can extract segments which does not have a specific appearance, but have only a semantic interpretation defined by a user.

Table 4.2 presents mean prediction error over 45 pairs of camel images (we exclude comparison of identical images). Each pair has two associated scores: prediction error of the first image given the second one and vice versa. We have thus 90 scores for each algorithm which are used to compute means and standard deviations. Like in

the previous sections, graph matching algorithms are run using information on vertex coordinates (using Eq. (4.3)), with $C_{ij} = e^{-(x_i-x_j)^2-(y_i-y_j)^2}$. Parameter λ in (4.3) as well as k (number of neighbors in KNN classifier) are learned via cross-validation. Here, again we observe the the "Grad" algorithm works better than other methods.

Table 4.2: Identification of object composite parts. (IE, STD)—mean and standard deviation of prediction error, for more explanations see text. Note, STD is not divided by the square root of the sample size.

	Grad	Spec	Beam	One-to-one
IE	0.303	0.351	0.432	0.342
STD	0.135	0.095	0.092	0.094

4.6 Conclusion

The main contribution of this chapter is the new formulation of the many-to-many graph matching problem as a discrete optimization problem (4.4) and the approximate algorithm "Grad" based on continious relaxation of (4.4) (Section 4.3). The success of the proposed method compared to other competitive approaches may be explained by two reasons. Methods based on continuous relaxations of discrete optimization problems often show a better perfomance than local search algorithm due to their ability to better explore the optimization set. At the same time the existing spectral approach for the many-to-many case [Caelli and Kosinov, 2004] is a formal extension of the one-to-one spectral method. This extension is "intuitive" but it does not have an optimization background as the one-to-one spectral algorithm [Umeyama, 1988], while the "Grad" algorithm aims to optimize a clear objective function naturally representing the quality of graph matching.

Besides a natural application of graph matching as a similarity measure between objects with complex structure, graph matching may be also used for object alignment. The second aspect is much less studied but we believe it may have interesting applications. For instance, in Section 4.5.3 we show how graph matching may be used for semantic segmentation based not on image morphology but on user-defined concepts.

In future work, we plan to complete the used datset of camel images by other images from the MPEG7 database, this work involves manual segmentation of all images and it is time consuming but it will be very interesting to test all methods on a larger dataset with more varied objects.

Part II

Other applications

Chapter 5

Phrase-Based Statistical Machine Translation as a Traveling Salesman Problem

Abstract

An efficient decoding algorithm is a crucial element of any statistical machine translation system. Some researchers have noted certain similarities between SMT decoding and the famous Traveling Salesman Problem; in particular Knight [1999] has shown that any TSP instance can be mapped to a sub-case of a word-based SMT model, demonstrating NP-hardness of the decoding task. In this chapter, we focus on the reverse mapping, showing that any phrase-based SMT decoding problem can be directly reformulated as a TSP. The transformation is very natural, deepens our understanding of the decoding problem, and allows direct use of any of the powerful existing TSP solvers for SMT decoding. We test our approach on three datasets, and compare a TSP-based decoder to the popular beam-search algorithm. In all cases, our method provides competitive or better performance.

This work was conducted during my internship at Xerox Research Center Europe with Marc Dymetman and Nicola Cancedda. This chapter is a slightly modified version of [Zaslavskiy et al., 2009].

In the very beginning we were interested in the application of graph matching to machine translation, but after several weeks of research it became clear that graph matching algorithms are too general for this purpose. It is possible to reformulate the machine translation problem as a graph matching problem, but this reformulation is rather artificial since it uses the TSP (traveling salesman problem) formulation as an intermediate step. The traveling salesman problem can be reduced to the graph matching problem (tsp corresponds to the alignment of a linear graph with the TSP graph), but, obviously, algorithms designed for TSP are more efficient for TSP than general graph matching algorithms. So we decided to keep the TSP formulation.

1 Introduction

Phrase-based systems [Koehn et al., 2003] are probably the most widespread class of Statistical Machine Translation systems, and arguably one of the most successful. They use aligned sequences of words, called biphrases, as building blocks for translations, and score alternative candidate translations for the same source sentence based on a log-linear model of the conditional probability of target sentences given the source sentence:

$$p(T, a|S) = \frac{1}{Z_S} \exp \sum_k \lambda_k h_k(S, a, T) \tag{5.1}$$

where the h_k are features, that is, functions of the source string S, of the target string T, and of the alignment a, where the alignment is a representation of the sequence of biphrases that where used in order to build T from S; The λ_k's are weights and Z_S is a normalization factor that guarantees that p is a proper conditional probability distribution over the pairs (T, A). Some features are *local*, i.e. decompose over biphrases and can be precomputed and stored in advance. These typically include forward and reverse phrase conditional probability features $\log p(\tilde{t}|\tilde{s})$ as well as $\log p(\tilde{s}|\tilde{t})$, where $\tilde{s}$ is the source side of the biphrase and $\tilde{t}$ the target side, and the so-called "phrase penalty" and "word penalty" features, which count the number of phrases and words in the alignment. Other features are *non-local*, i.e. depend on the order in which

biphrases appear in the alignment. Typical non-local features include one or more n-gram language models as well as a distortion feature, measuring by how much the order of biphrases in the candidate translation deviates from their order in the source sentence.

Given such a model, where the λ_i's have been tuned on a development set in order to minimize some error rate (see e.g. [Lopez, 2008]), together with a library of biphrases extracted from some large training corpus, a *decoder* implements the actual search among alternative translations:

$$(a^*, T^*) = \arg\max_{(a,T)} P(T, a|S). \tag{5.2}$$

The decoding problem (5.2) is a discrete optimization problem. Usually, it is very hard to find the exact optimum and, therefore, an approximate solution is used. Currently, most decoders are based on some variant of a heuristic left-to-right search, that is, they attempt to build a candidate translation (a, T) incrementally, from left to right, extending the current partial translation at each step with a new biphrase, and computing a score composed of two contributions: one for the known elements of the partial translation so far, and one a heuristic estimate of the remaining cost for completing the translation. The variant which is mostly used is a form of *beam-search*, where several partial candidates are maintained in parallel, and candidates for which the current score is too low are pruned in favor of candidates that are more promising.

We will see in the next section that some characteristics of beam-search make it a suboptimal choice for phrase-based decoding, and we will propose an alternative. This alternative is based on the observation that phrase-based decoding can be very naturally cast as a Traveling Salesman Problem (TSP), one of the best studied problems in combinatorial optimization. We will show that this formulation is not only a powerful conceptual device for reasoning on decoding, but is also practically convenient: in the same amount of time, off-the-shelf TSP solvers can find higher scoring solutions than the state-of-the art beam-search decoder implemented in *Moses* [Hoang and Koehn, 2008].

2 Related work

Beam-search decoding

In beam-search decoding, candidate translation prefixes are iteratively extended with new phrases. In its most widespread variant, *stack decoding*, prefixes obtained by consuming the same number of source words, no matter which, are grouped together in the same *stack*[1] and compete against one another. *Threshold* and *histogram* pruning are applied: the former consists in dropping all prefixes having a score lesser than the best score by more than some fixed amount (a parameter of the algorithm), the latter consists in dropping all prefixes below a certain rank.

While quite successful in practice, stack decoding presents some shortcomings. A first one is that prefixes obtained by translating different subsets of source words compete against one another. In one early formulation of stack decoding for SMT [Germann et al., 2001], the authors indeed proposed to lazily create one stack for each subset of source words, but acknowledged issues with the potential combinatorial explosion in the number of stacks. This problem is reduced by the use of heuristics for estimating the cost of translating the remaining part of the source sentence. However, this solution is only partially satisfactory. On the one hand, heuristics should be computationally light, much lighter than computing the actual best score itself, while, on the other hand, the heuristics should be tight, as otherwise pruning errors will ensue. There is no clear criterion to guide in this trade-off. Even when good heuristics are available, the decoder will show a bias towards putting at the beginning the translation of a certain portion of the source, either because this portion is less ambiguous (i.e. its translation has larger conditional probability) or because the associated heuristics is less tight, hence more optimistic. Finally, since the translation is built left-to-right the decoder cannot optimize the search by taking advantage of highly unambiguous and informative portions that should be best translated far from the beginning. All these reasons motivate considering alternative decoding strategies.

Word-based SMT and the TSP

[1] While commonly adopted in the speech and SMT communities, this is a bit of a misnomer, since the used data structures are priority queues, not stacks.

As already mentioned, the similarity between SMT decoding and TSP was recognized in [Knight, 1999], who focussed on showing that any TSP can be reformulated as a sub-class of the SMT decoding problem, proving that SMT decoding is NP-hard. Following this work, the existence of many efficient TSP algorithms then inspired certain adaptations of the underlying techniques to SMT decoding for word-based models. Thus, Germann et al. [2001] adapt a TSP subtour elimination strategy to an IBM-4 model, using generic Integer Programming techniques. The chapter comes close to a TSP formulation of decoding with IBM-4 models, but does not pursue this route to the end, stating that *"It is difficult to convert decoding into straight TSP, but a wide range of combinatorial optimization problems (including TSP) can be expressed in the more general framework of linear integer programming"*. By employing generic IP techniques, it is however impossible to rely on the variety of more efficient both exact and approximate approaches which have been designed specifically for the TSP. In [Tillmann and Ney, 2003] and [Tillmann, 2006], the authors modify a certain Dynamic Programming technique used for TSP for use with an IBM-4 word-based model and a phrase-based model respectively. However, to our knowledge, none of these works has proposed a direct reformulation of these SMT models as TSP instances. We believe we are the first to do so, working in our case with the mainstream phrase-based SMT models, and therefore making it possible to directly apply existing TSP solvers to SMT.

3 The Traveling Salesman Problem and its variants

In this paper the Traveling Salesman Problem appears in four variants:

STSP. The most standard, and most studied, variant is the *Symmetric TSP*: we are given a non-directed graph G on N nodes, where the edges carry real-valued costs. The STSP problem consists in finding a tour of minimal total cost, where a tour (also called Hamiltonian Circuit) is a "circular" sequence of nodes visiting each node of the graph exactly once;

ATSP. The *Asymmetric TSP*, or ATSP, is a variant where the underlying graph G is directed and where, for i and j two nodes of the graph, the edges (i,j) and (j,i) may carry different costs.

SGTSP. The *Symmetric Generalized TSP*, or SGTSP: given a non-oriented graph G of $|G|$ nodes with edges carrying real-valued costs, given a partition of these $|G|$ nodes into m non-empty, disjoint, subsets (called clusters), find a circular sequence of m nodes of minimal total cost, where each cluster is visited exactly once.

AGTSP. The *Asymmetric Generalized TSP*, or AGTSP: similar to the SGTSP, but G is now a directed graph.

The STSP is often simply denoted TSP in the literature, and is known to be NP-hard [Applegate et al., 2007]; however there has been enormous interest in developing efficient solvers for it, both exact and approximate.

Most of existing algorithms are designed for *STSP*, but *ATSP*, *SGTSP* and *AGTSP* may be reduced to *STSP*, and therefore solved by *STSP* algorithms.

3.1 Reductions AGTSP→ATSP→STSP

The transformation of the AGTSP into the ATSP, introduced by Noon and Bean [1993]), is illustrated in Figure (5.1). In this diagram, we assume that $Y_1, \ldots, Y_K$ are the nodes of a given cluster, while X and Z are arbitrary nodes belonging to other clusters. In the transformed graph, we introduce edges between the Y_i's in order to form a cycle as shown in the figure, where each edge has a large negative cost $-K$. We leave alone the incoming edge to Y_i from X, but the outgoing edge from Y_i to X has its origin changed to Y_{i-1}. A feasible tour in the original AGTSP problem passing through X, Y_i, Z will then be "encoded" as a tour of the transformed graph that first traverses X , then traverses $Y_i, \ldots, Y_K, \ldots, Y_{i-1}$, then traverses Z (this encoding will have the same cost as the original cost, minus $(k-1)K$). Crucially, if K is large enough, then the solver for the transformed ATSP graph will tend to traverse as many K edges as possible, meaning that it will traverse exactly $k-1$ such edges in the cluster, that is, it will produce an encoding of some feasible tour of the AGTSP problem.

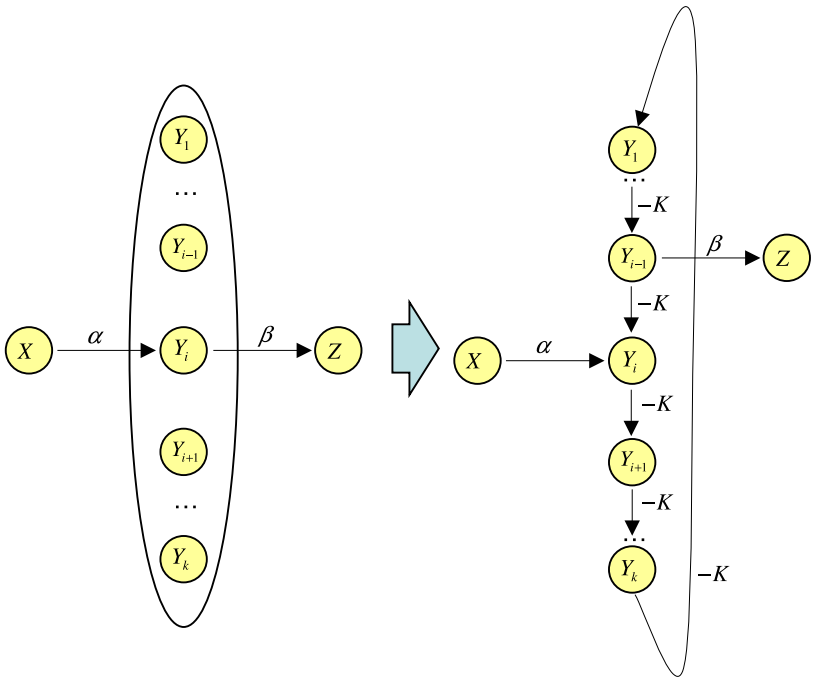


Figure 5.1: AGTSP→ATSP.

As for the transformation ATSP→STSP, several variants are described in the literature, e.g. [Applegate et al., 2007, p. 126]; the one we use is from [Wikipedia, 2009] (not illustrated here for lack of space).

3.2 TSP algorithms

TSP is one of the most studied problems in combinatorial optimization, and even a brief review of existing approaches would take too much place. Interested readers may consult [Applegate et al., 2007; Gutin, 2003] for good introductions.

One of the best existing TSP solvers is implemented in the open source *Concorde* package [Applegate et al., 2005]. *Concorde* includes the fastest exact algorithm and one of the most efficient implementations of the Lin-Kernighan (LK) heuristic for finding an approximate solution. LK works by generating an initial random feasible solution for the TSP problem, and then repeatedly identifying an ordered subset of k edges in the current tour and an ordered subset of k edges not included in the

tour such that when they are swapped the objective function is improved. This is somewhat reminiscent of the *Greedy decoding* of Germann et al. [2001], but in LK several transformations can be applied simultaneously, so that the risk of being stuck in a local optimum is reduced [Applegate et al., 2007, chapter 15].

As will be shown in the next section, phrase-based SMT decoding can be directly reformulated as an AGTSP. Here we use *Concorde* through first transforming AGTSP into STSP, but it might also be interesting in the future to use algorithms specifically designed for AGTSP, which could improve efficiency further (see Conclusion).

4 Phrase-based Decoding as TSP

In this section we reformulate the SMT decoding problem as an *AGTSP*. We will illustrate the approach through a simple example: translating the French sentence *"cette traduction automatique est curieuse"* into English. We assume that the relevant biphrases for translating the sentence are as follows:

ID	source	target
h	*cette*	*this*
t	*traduction*	*translation*
ht	*cette traduction*	*this translation*
mt	*traduction automatique*	*machine translation*
a	*automatique*	*automatic*
m	*automatique*	*machine*
i	*est*	*is*
s	*curieuse*	*strange*
c	*curieuse*	*curious*

Under this model, we can produce, among others, the following translations:

$h \cdot mt \cdot i \cdot s$	*this machine translation is strange*
$h \cdot c \cdot t \cdot i \cdot a$	*this curious translation is automatic*
$ht \cdot s \cdot i \cdot a$	*this translation strange is automatic*

where we have indicated on the left the ordered sequence of biphrases that leads to each translation.

We now formulate decoding as an AGTSP, in the following way. The graph nodes are all the possible pairs (w, b), where w is a source word in the source sentence s and b is a biphrase containing this source word. The graph clusters are the subsets of the graph nodes that share a common source word w.

The costs of a transition between nodes M and N of the graph are defined as follows:

(a) If M is of the form (w, b) and N of the form (w', b), in which b is a single biphrase, and w and w' are consecutive words in b, then the transition cost is 0: once we commit to using the first word of b, there is no additional cost for traversing the other source words covered by b.

(b) If $M = (w, b)$, where w is the *rightmost source word* in the biphrase b, and $N = (w', b')$, where $w' \neq w$ is the *leftmost source word* in b', then the transition cost corresponds to the cost of selecting b' just after b; this will correspond to "consuming" the source side of b' after having consumed the source side of b (whatever their relative positions in the source sentence), and to producing the target side of b' directly after the target side of b; the transition cost is then the addition of several contributions (weighted by their respective λ (not shown), as in equation 5.1):

- The cost associated with the features local to b in the biphrase library;
- The "distortion" cost of consuming the source word w' just after the source word w: $|\text{pos}(w') - \text{pos}(w) - 1|$, where $\text{pos}(w)$ and $\text{pos}(w')$ are the positions of w and w' in the source sentence.
- The language model cost of producing the target words of b' right after the target words of b; with a bigram language model, this cost can be precomputed directly from b and b'. This restriction to bigram models will be removed in Section 4.1.

(c) In all other cases, the transition cost is infinite, or, in other words, there is no edge in the graph between M and N.

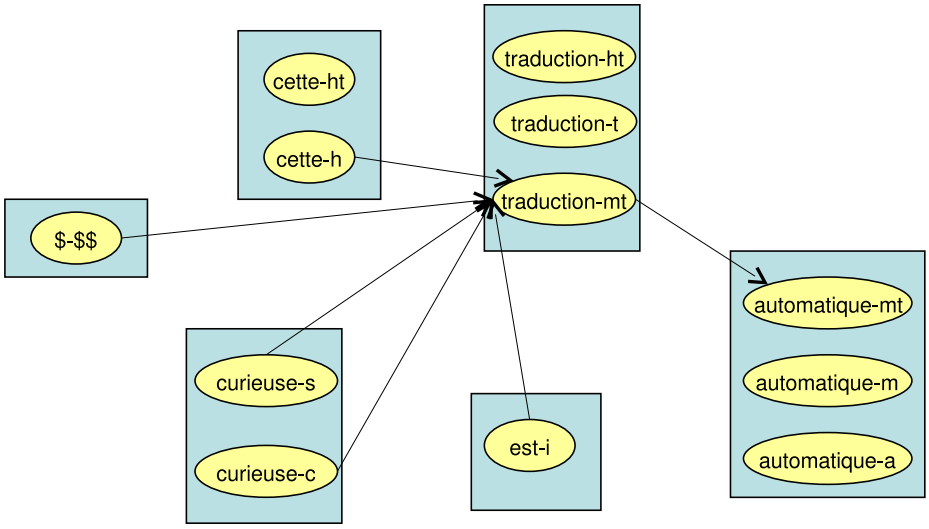


Figure 5.2: Transition graph for the source sentence *cette traduction automatique est curieuse*. Only edges entering or exiting the node traduction − mt are shown. The only successor to [traduction − mt] is [automatique − mt], and [cette − ht] is not a predecessor of [traduction − mt].

A special cluster containing a single node (denoted by $-$$ in the figures), and corresponding to special *beginning-of-sentence* symbols must also be included: the corresponding edges and weights can be worked out easily. Figures 5.2 and 5.3 give some illustrations of what we have just described.

4.1 From Bigram to N-gram LM

Successful phrase-based systems typically employ language models of order higher than two. However, our models so far have the following important "Markovian" property: the cost of a path is additive relative to the costs of transitions. For example, in the example of Figure 5.3, the cost of *this* · *machine translation* · *is* · *strange*, can only take into account the conditional probability of the word *strange* relative to the word *is*, but not relative to the words *translation* and *is*. If we want to extend the power of the model to general n-gram language models, and in particular to the 3-gram case (on which we concentrate here, but the techniques can be easily extended to the general case), the following approach can be applied.

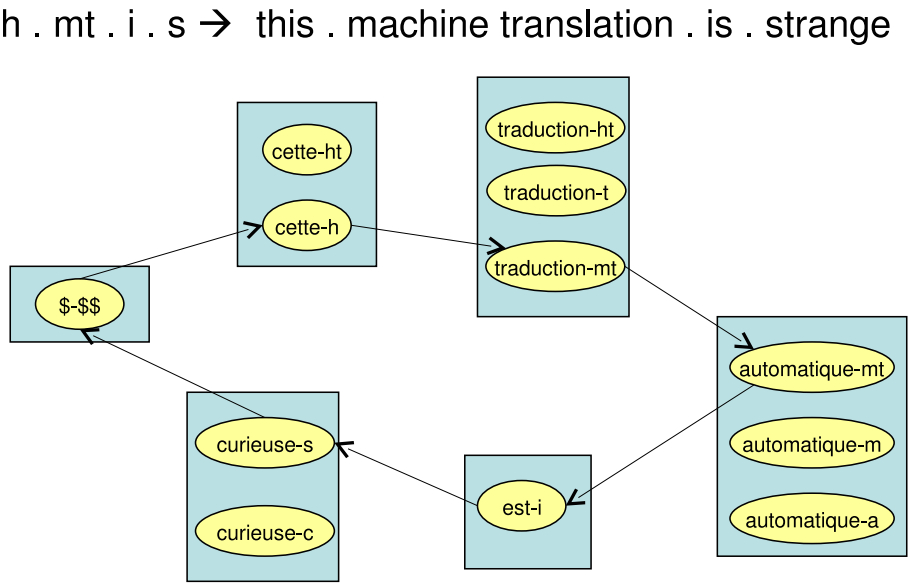


Figure 5.3: A GTSP tour is illustrated, corresponding to the displayed output.

Compiling Out for Trigram models

This approach consists in "compiling out" all biphrases with a target side of only one word. We replace each biphrase b with single-word target side by "extended" biphrases $b_1, \ldots, b_r$, which are "concatenations" of b and some other biphrase b' in the library.[2] To give an example, consider that we: (1) remove from the biphrase library the biphrase i, which has a single word target, and (2) add to the library the extended biphrases *mti*, *ti*, *si*, ..., that is, all the extended biphrases consisting of the concatenation of a biphrase in the library with i, then it is clear that these extended biphrases will provide enough context to compute a trigram probability for the target word produced immediately next (in the examples, for the words *strange*, *automatic* and *automatic* respectively). If we do that exhaustively for all biphrases (relevant for the source sentence at hand) that, like i, have a single-word target, we will obtain a representation that allows a trigram language model to be computed at each point.

[2]In the figures, such "concatenations" are denoted by $[b' \cdot b]$; they are interpreted as encapsulations of first consuming the source side of b', *whether or not this source side precedes the source side of* b *in the source sentence*, producing the target side of b', consuming the source side of b, and producing the target side of b immediately after that of b'.

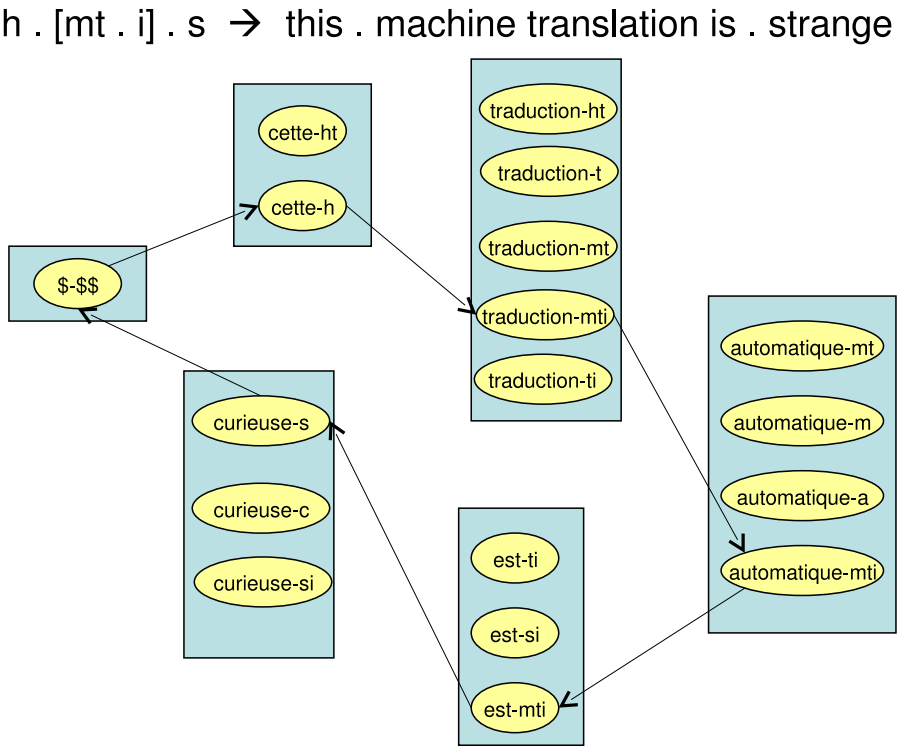


Figure 5.4: Compiling-out of biphrase i: (est,is).

The situation becomes clearer by looking at Figure 5.4, where we have only eliminated the biphrase i, and only shown some of the extended biphrases that now encapsulate i, and where we show one valid circuit. Note that we are now able to associate with the edge connecting the two nodes (est, mti) and (curieuse, s) a trigram cost because mti provides a large enough target context.

While this exhaustive "compiling out" method works in principle, it has a serious defect: if for the sentence to be translated, there are m relevant biphrases, among which k have single-word targets, then we will create on the order of km extended biphrases, which may represent a significant overhead for the TSP solver, as soon as k is large relative to m, which is typically the case. The problem becomes even worse if we extend the compiling-out method to n-gram language models with $n > 3$. In the Future Work section below, we describe a powerful approach for circumventing this problem, but with which we have not experimented yet.

5 Experiments

5.1 Monolingual word re-ordering

In the first series of experiments we consider the artificial task of reconstructing the original word order of a given English sentence. First, we randomly permute words in the sentence, and then we try to reconstruct the original order by maximizing the LM score over all possible permutations. The reconstruction procedure may be seen as a translation problem from "Bad English" to "Good English". Usually the LM score is used as one component of a more complex decoder score which also includes biphrase and distortion scores. But in this particular "translation task" from bad to good English, we consider that all "biphrases" are of the form $e - e$, where e is an English word, and we do not take into account any distortion: we only consider the quality of the permutation as it is measured by the LM component. Since for each "source word" e, there is exactly one possible "biphrase" $e - e$ each cluster of the Generalized TSP representation of the decoding problem contains exactly one node; in other terms, the Generalized TSP in this situation is simply a standard TSP. Since the decoding phase is then equivalent to a word reordering, the LM score may be used to compare the performance of different decoding algorithms. Here, we compare three different algorithms: classical beam-search (*Moses*); a decoder based on an exact TSP solver (*Concorde*); a decoder based on an approximate TSP solver (Lin-Kernighan as implemented in the Concorde solver) [3]. In the Beam-search and the LK-based TSP solver we can control the trade-off between approximation quality and running time. To measure re-ordering quality, we use two scores. The first one is just the "internal" LM score; since all three algorithms attempt to maximize this score, a natural evaluation procedure is to plot its value versus the elapsed time. The second score is BLEU [Papineni et al., 2001], computed between the reconstructed and the original sentences, which allows us to check how well the quality of reconstruction correlates with the internal score. The training dataset for learning the LM consists of 50000 sentences from NewsCommentary corpus [Callison-Burch et al., 2008], the

[3] Both TSP decoders may be used with/or without a *distortion limit*; in our experiments we do not use this parameter.

test dataset for word reordering consists of 170 sentences, the average length of test sentences is equal to 17 words.

Bigram based reordering. First we consider a bigram Language Model and the algorithms try to find the re-ordering that maximizes the LM score. The TSP solver used here is exact, that is, it actually finds the optimal tour. Figures 5.5(a,b) present the performance of the TSP and Beam-search based methods.

Trigram based reordering. Then we consider a trigram based Language Model and the algorithms again try to maximize the LM score. The trigram model used is a variant of the exhaustive compiling-out procedure described in Section 4.1. Again, we use an exact TSP solver.

Looking at Figure 5.5a, we see a somewhat surprising fact: the cross and some star points have positive y coordinates! This means that, when using a bigram language model, it is often possible to reorder the words of a randomly permuted reference sentence in such a way that the LM score of the reordered sentence is larger than the LM of the reference. A second notable point is that the increase in the LM-score of the beam-search with time is steady but very slow, and never reaches the level of performance obtained with the exact-TSP procedure, even when increasing the time by several orders of magnitude. Also to be noted is that the solution obtained by the exact-TSP is provably the optimum, which is almost never the case of the beam-search procedure. In Figure 5.5b, we report the BLEU score of the reordered sentences in the test set relative to the original reference sentences. Here we see that the exact-TSP outputs are closer to the references in terms of BLEU than the beam-search solutions. Although the TSP output does not recover the reference sentences (it produces sentences with a slightly higher LM score than the references), it does reconstruct the references better than the beam-search. The experiments with trigram language models (Figures 5.5(c,d)) show similar trends to those with bigrams.

5.2 Translation experiments with a bigram language model

In this section we consider two real translation tasks, namely, translation from English to French, trained on Europarl [Koehn et al., 2003] and translation from German to

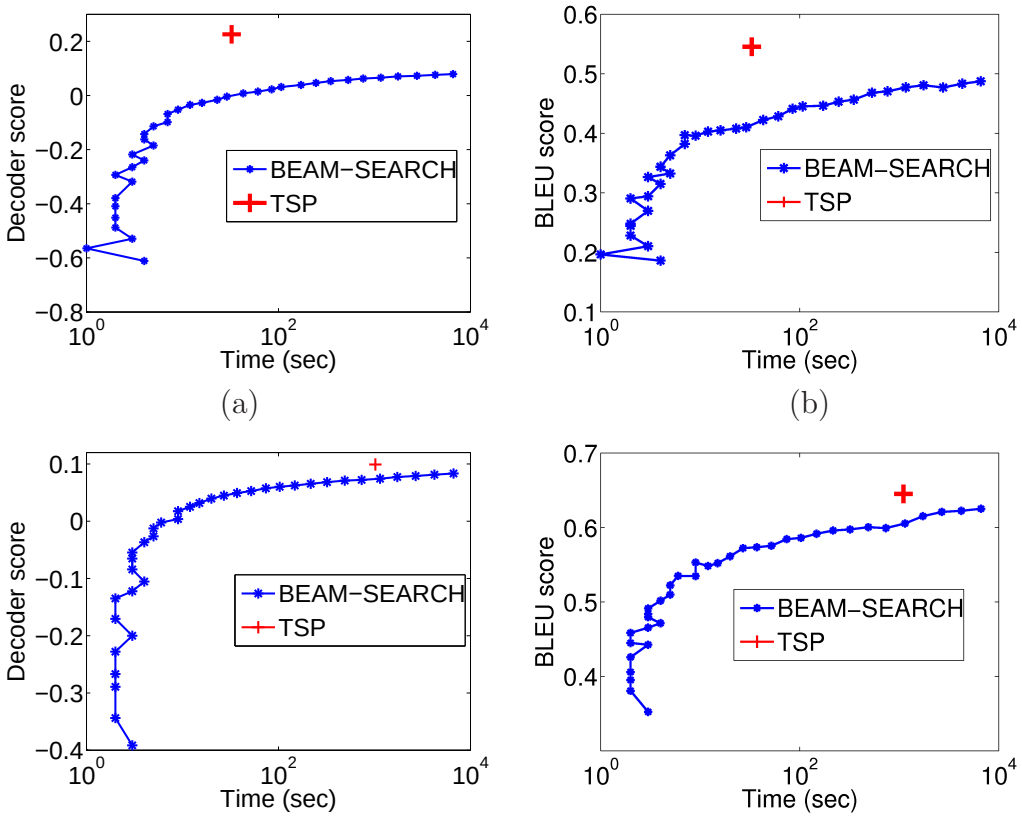


Figure 5.5: (a), (b): LM and BLEU scores as functions of time for a bigram LM; (c), (d): the same for a trigram LM. The x axis corresponds to the cumulative time for processing the test set; for (a) and (c), the y axis corresponds to the mean difference (over all sentences) between the lm score of the output and the lm score of the reference normalized by the sentence length N: (LM(ref)-LM(true))/N. The solid line with star marks corresponds to using beam-search with different pruning thresholds, which result in different processing times and performances. The cross corresponds to using the exact-TSP decoder (in this case the time to the optimal solution is not under the user's control).

Spanish training on the NewsCommentary corpus. For Europarl, the training set includes 2.81 million sentences, and the test set 500. For NewsCommentary the training set is smaller: around 63k sentences, with a test set of 500 sentences. Figure 5.6 presents Decoder and Bleu scores as functions of time for the two corpuses.

Since in the real translation task, the size of the TSP graph is much larger than in the artificial reordering task (in our experiments the median size of the TSP graph was around 400 nodes, sometimes growing up to 2000 nodes), directly applying the exact TSP solver would take too long; instead we use the approximate LK algorithm and compare it to Beam-Search. The efficiency of the LK algorithm can be significantly increased by using a good initialization. To compare the quality of the LK and Beam-Search methods we take a rough initial solution produced by the Beam-Search algorithm using a small value for the stack size and then use it as initial point, both for the LK algorithm and for further Beam-Search optimization (where as before we vary the Beam-Search thresholds in order to trade quality for time).

In the case of the Europarl corpus, we observe that LK outperforms Beam-Search in terms of the Decoder score as well as in terms of the BLEU score. Note that the difference between the two algorithms increases steeply at the beginning, which means that we can significantly increase the quality of the Beam-Search solution by using the LK algorithm at a very small price. In addition, it is important to note that the BLEU scores obtained in these experiments correspond to feature weights, in the log-linear model (5.1), that have been optimized for the Moses decoder, but not for the TSP decoder: optimizing these parameters relatively to the TSP decoder could improve its BLEU scores still further.

On the News corpus, again, LK outperforms Beam-Search in terms of the Decoder score. The situation with the BLEU score is more confuse. Both algorithms do not show any clear score improvement with increasing running time which suggests that the decoder's objective function is not very well correlated with the BLEU score on this corpus.

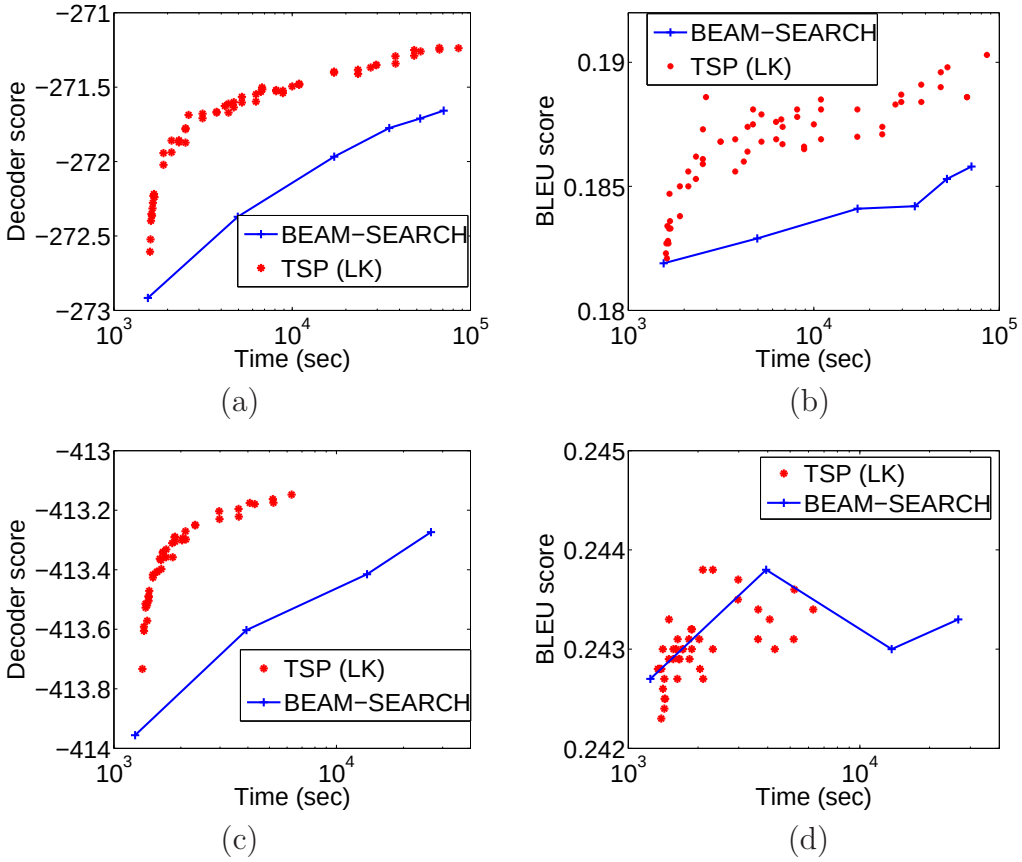


Figure 5.6: (a), (b): Europarl corpus, translation from English to French; (c),(d): NewsCommentary corpus, translation from German to Spanish. Average value of the decoder and the BLEU scores (over 500 test sentences) as a function of time. The trade-off quality/time in the case of LK is controlled by the number of iterations, and each point corresponds to a particular number of iterations, in our experiments LK was run with a number of iterations varying between 2k and 170k. The same trade-off in the case of Beam-Search is controlled by varying the beam thresholds.

6 Future Work

In section 4.1, we described a general "compiling out" method for extending our TSP representation to handling trigram and N-gram language models, but we noted that the method may lead to combinatorial explosion of the TSP graph. While this problem was manageable for the artificial monolingual word re-ordering (which had only one possible translation for each source word), it becomes unwieldy for the real translation experiments, which is why in this chapter we only considered bigram LMs for these experiments. However, we know how to handle this problem in principle, and we now describe a method that we plan to experiment with in the future.

To avoid the large number of artificial biphrases as in 4.1, we perform an *adaptive selection*. Let us suppose that (w, b) is a SMT decoding graph node, where b is a biphrase containing only one word on the target side. On the first step, when we evaluate the traveling cost from (w, b) to (w', b'), we take the language model component equal to

$$\min_{b'' \neq b', b} -\log p(b'.v | b.e, b''.e),$$

where $b'.v$ represents the first word of the b' target side, $b.e$ is the only word of the b target side, and $b''.e$ is the last word of the b'' target size. This procedure underestimates the total cost of tour passing through biphrases that have a single-word target. Therefore if the optimal tour passes only through biphrases with more than one word on their target side, then we are sure that this tour is also optimal in terms of the tri-gram language model. Otherwise, if the optimal tour passes through (w, b), where b is a biphrase having a single-word target, we add only the extended biphrases related to b as we described in section 4.1, and then we recompute the optimal tour. Iterating this procedure provably converges to an optimal solution.

This powerful method, which was proposed in [Kam and Kopec, 1996; Popat et al., 2001] in the context of a finite-state model (but not of TSP), can be easily extended to N-gram situations, and typically converges in a small number of iterations.

7 Conclusion

The main contribution of this chapter has been to propose a transformation for an arbitrary phrase-based SMT decoding instance into a TSP instance. While certain similarities of SMT decoding and TSP were already pointed out in [Knight, 1999], where it was shown that any Traveling Salesman Problem may be reformulated as an instance of a (simplistic) SMT decoding task, and while certain techniques used for TSP were then adapted to word-based SMT decoding [Germann et al., 2001; Tillmann and Ney, 2003; Tillmann, 2006], we are not aware of any previous work that shows that SMT decoding can be directly reformulated as a TSP. Beside the general interest of this transformation for understanding decoding, it also opens the door to direct application of the variety of existing TSP algorithms to SMT. Our experiments on synthetic and real data show that fast TSP algorithms can handle selection and reordering in SMT comparably or better than the state-of-the-art beam-search strategy, converging on solutions with higher objective function in a shorter time.

The proposed method proceeds by first constructing an AGTSP instance from the decoding problem, and then converting this instance first into ATSP and finally into STSP. At this point, a direct application of the well known STSP solver *Concorde* (with Lin-Kernighan heuristic) already gives good results. We believe however that there might exist even more efficient alternatives. Instead of converting the AGTSP instance into a STSP instance, it might prove better to use directly algorithms expressly designed for ATSP or AGTSP. For instance, some of the algorithms tested in the context of the *DIMACS* implementation challenge for ATSP [Johnson et al., 2002] might well prove superior. There is also active research around AGTSP algorithms. Recently new effective methods based on a "memetic" strategy [Buriol et al., 2004; Gutin et al., 2008] have been put forward. These methods combined with our proposed formulation provide ready-to-use SMT decoders, which it will be interesting to compare.

Acknowledgments

Thanks to Vassilina Nikoulina for her advice about running Moses on the test datasets.

Chapter 6

A new protein binding pocket similarity measure based on comparison of 3D atom clouds: application to ligand prediction

Abstract

Prediction of ligands for proteins of known 3D structure is important to understand structure-function relationship, predict molecular function, or design new drugs. We explore a new approach for ligand prediction in which binding pockets are represented by atom clouds. Each target pocket is compared to an ensemble of pockets of known ligands. Pockets are aligned in 3D space with further use of convolution kernels between clouds of points. Performance of the new method for ligand prediction is compared to those of other available measures and to docking programs. We discuss two criteria to compare the quality of similarity measures: area under ROC curve (AUC) and classification based scores. We show that the latter is better suited to evaluate the methods with respect to ligand prediction. Our results on existing and new benchmarks indicate that the new method outperforms other approaches, including docking.

This project was done in collaboration with Brice Hoffmann and Veronique Stoven for the Center for Computantional Biology, Mines-ParisTech.

1 Introduction

One of the main goals of structural biology is to predict, from the 3D fold of a protein, its interacting partners, which in turn is related to its molecular function. However, understanding this structure-function relationship is still today an open question, and no reliable tool is available to permit such a prediction. Current efforts concentrate on local 3D approaches, focusing on identification and comparison of binding pockets, in order to predict the natural ligand for a protein, with the underlying idea that proteins sharing similar binding sites are expected to bind similar ligands. The same strategy also applies to the problem of identifying new drug precursors for a therapeutic target protein.

The comparison of 3D binding pockets is an active field of research, and during the last decade, many new methods were proposed. Morris et al. [2005] considered a method based on using real spherical harmonic expansion coefficients, Gold and Jackson [2006] used a specialized geometric hashing procedure as the core of the SitesBase web server, Shulman-Peleg et al. [2008] used multiple common point set detection method. An approach proposed by Schalon et al. [2008] is based on a triangle-discretized sphere representation of binding pockets. Weskamp et al. [2007] and Najmanovich et al. [2008] considered graph-based representations of binding pockets and applied graph matching algorithms.

In this chapter, we explore the potential of a new approach in which binding pockets are represented by clouds of atoms in 3D space potentially baring additional labels such as partial charge or atom type. The new similarity measure is based on the alignment of protein pockets with further use of convolution kernel between 3D point clouds. We study how the proposed method may be used to predict a ligand for a given pocket by comparing it to a set of pockets with known ligand.

Here, we do not discuss the problem of pocket detection. In our experiments, we extracted pockets on the basis of known protein-ligand crystal structures as it was

done by Kahraman et al. [2007]. In cases where the binding site is unknown, various programs have been developed to locate depressions on protein surfaces and could be used to identify putative binding sites [Glaser et al., 2006].

An important question in this chapter is the evaluation of pocket similarity measures. We discuss two criteria to compare the quality of similarity measures on the basis of their ability to detect pockets binding the same ligand: area under ROC curve (AUC) and classification based scores. We compare our method with some existing state of the art algorithms on different benchmark datasets. Since we evaluate methods for binding pocket comparison according to their ability to predict ligands, we also report the performance of docking methods, on the same benchmark datasets. Finally, we also discuss possible extensions of the proposed method to other applications such as protein function prediction or ligand comparison.

2 Methods

2.1 Convolution kernel between atom clouds

In our model, a binding pocket is described by a set of atoms in 3D space. Our objective is to construct a similarity measure between pockets, which may be used to identify pockets binding the same ligand.

Let $P = (x_i, l_i)_{i=1}^N$ denote a binding pocket consisting of N atoms, where $x_i \in \mathbb{R}^3$ is a 3D vector representing atom coordinates, and l_i is a label (discrete or real valued) that may be used to bare additional information on the atoms (for example, atom type, atom partial charge, or amino acid type).

A classical approach for pocket comparison consists in iterative alignment of two pockets and further counting of overlapping atoms, usually within a tolerance of 1Å. Different implementations of this principle may be found in such methods as Tanimoto index [Willett et al., 1986], the SitesBase algorithm (Poisson index), or the MultiBind algorithm [Shulman-Peleg et al., 2008]. The alignment is made to maximize the number of overlapping atoms, which is generally a good indicator of pocket similarity.

However, atoms may have different positions but play equivalent roles in ligand binding, and the role of one atom in one pocket may be played by a group of atoms in another one. These observations lead us to the idea of an alternative smooth score which does not count the number of overlapping atoms, but rather uses a weighted number of atoms having closed positions. We first consider the case where labels are ignored, and only atom coordinates are used to measure the similarity between pockets, and then explain how the information on atom labels may be introduced in the new similarity measure.

Given two pockets P_1 and P_2 the similarity measure $K(P_1, P_2)$ is defined as follows

$$K(P_1, P_2) = \sum_{x_i \in P_1} \sum_{y_j \in P_2} e^{\frac{-||x_i - y_j||^2}{2\sigma^2}}. \tag{6.1}$$

This similarity measure defines in fact a positive definite kernel, i.e. it may be considered as a true scalar product on the set of atom clouds representing binding pockets [Schölkopf et al., 2004]. Implicitly, it defines a distance between pockets: $D(P_1, P_2) = K(P_1, P_1) + K(P_2, P_2) - 2K(P_1, P_2)$ which has all standard properties of a true metric (non-negativity, identity of indiscernibles, symmetry, triangular inequality). The parameter σ characterizes the sensitivity of the similarity measure (6.1) to points relative displacements. When σ is small, only atoms of two pockets which are very close to each other significantly contribute to $K(P_1, P_2)$. On the contrary, when σ is large, almost all pairs of atoms contribute to $K(P_1, P_2)$.

The kernel (6.1) is an example of a convolution kernel [Haussler, 1999; Gärtner et al., 2002] between point sets. Alternative kernels may be constructed by substituting the Gaussian kernel $e^{\frac{-||x_i - y_j||^2}{2\sigma^2}}$ by any other kernel between 3D vectors x_i and y_j.

Interestingly, the kernel (6.1) may be seen as a particular case of kernel between point sets defined as a kernel between distribution function estimated from point sets [Kondor and Jebara, 2003]. More precisely, let us represent each binding pocket P_i by a distribution of masses defined as the sum of Gaussian with bandwidth $\sigma/\sqrt{2}$

functions centered on the pocket atoms, namely:

$$f_{P_i}(x) = \sum_{x_i \in P_i} e^{-\frac{||x-x_i||^2}{\sigma^2}}.$$

Then kernel (6.1) between pockets P_1 and P_2 can be recovered, up to a scaling constant, as the scalar product in $L_2(\mathbb{R}^3)$ between the associated distributions because:

$$\begin{aligned}\langle f_{P_1}, f_{P_2} \rangle_{L_2(\mathbb{R}^3)} &= \int_{\mathbb{R}^3} \sum_{x_i \in P_i} e^{-\frac{||x-x_i||^2}{\sigma^2}} \sum_{y_i \in P_j} e^{-\frac{||x-y_j||^2}{\sigma^2}} dx \\ = \sum_{\substack{x_i \in P_1 \\ y_j \in P_2}} \int_{\mathbb{R}^3} e^{-\frac{||x-x_i||^2}{\sigma^2}} e^{-\frac{||x-y_j||^2}{\sigma^2}} dx &= C \sum_{\substack{x_i \in P_1 \\ y_j \in P_2}} e^{-\frac{-||x_i-y_j||^2}{2\sigma^2}} \\ &= CK(P_1, P_2)\,,\end{aligned}$$

where C is a positive constant.

However, formula (6.1) is not fully appropriate in practice, because the proposed measure is not invariant upon rotations and translations of the binding pockets. Therefore, we define a similarity measure *sup-CK* as the maximum of (6.1) over all possible rotations and translations of one of the two pockets:

$$\text{sup-CK}(P_1, P_2) = \max_{R, y_t} \sum_{x_i \in P_1, y_j \in P_2} e^{\frac{||x_i - (Ry_j + y_t)||^2}{2\sigma^2}}, \tag{6.2}$$

where R is an orthonormal rotation matrix and y_t is a translation vector. *Sup-CK* is not a positive definite measure anymore, but it can still be used as a similarity score. Furthermore, to evaluate *sup-CK*, we now need to maximize a non-concave function over the set of rotations and translations, which may have many local maxima. Exact maximization of this non-concave function is a hard optimization problem and we propose to estimate an approximate solution by running a gradient ascent algorithm, starting from many different initial points, and taking the best local maximum. The optimization algorithm may be significantly accelerated by choosing an initial point close to the global optimum. In the case of binding pockets, a good approximation of the optimal translation vector y_t is the vector which translates the geometric center of

P_2 into the geometric center of P_1, $y_t = \frac{1}{N_1}\sum_{x_i \in P_1} x_i - \frac{1}{N_2}\sum_{y_i \in P_2} y_i$. The approximated rotation matrix R superposes the first principal axis of P_2 with the first principal axis of P_1, the second one with the second one, and the third one with the third one. Since principal vectors are defined up to a sign, the two signs for all principal vectors of one of the binding pockets have to be tested (there are 2^3 combinations). If some of the pocket axes have close lengths, then it may be also interesting to consider rotations which superpose the first principal axis of one pocket with the second principal axis of the other one.

Gradient ascent method requires to calculate the gradient of the function in (6.2) with respect to R and y_t. Calculation of the gradient components related to y_t is straightforward:

$$\nabla_{y_t} = \frac{1}{\sigma^2} \sum_{x_i \in P_1, y_j \in P_2} (x_i - (Ry_j + y_t)) e^{\frac{||x_i - (Ry_j + y_t)||^2}{2\sigma^2}}.$$

Since the set of rotation matrices is a 3D manifold embedded in 9D space, we cannot take derivatives with respect to each element of matrix R. Instead, we use the Euler representation of the rotation matrix:

$$\begin{aligned} R = R_X R_Y R_Z = &\begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\phi & \sin\phi \\ 0 & -\sin\phi & \cos\phi \end{bmatrix} \times \\ &\begin{bmatrix} \cos\theta & 0 & -\sin\theta \\ 0 & 1 & 0 \\ \sin\theta & 0 & \cos\theta \end{bmatrix} \begin{bmatrix} \cos\psi & \sin\psi & 0 \\ -\sin\psi & \cos\psi & 0 \\ 0 & 0 & 1 \end{bmatrix}, \end{aligned} \tag{6.3}$$

where R is expressed as a function of $(\phi, \theta, \psi) \in [0; 2\pi)^3$. The derivatives of the maximand in (6.2) are now calculated with respect to (ϕ, θ, ψ), for instance,

$$\nabla_\theta = \frac{1}{\sigma^2} \sum_{\substack{x_i \in P_1 \\ y_j \in P_2}} e^{\frac{||x_i - (Ry_j + y_t)||^2}{2\sigma^2}} (x_i - (Ry_j + y_t))^T \times \left(R_X \frac{\partial R_Y}{\partial \theta} R_Z y_j\right).$$

As mentioned above, it may be interesting to use additional information on binding pocket atoms (such as atom type or charge). Let us suppose that this information is represented by labels l_i (which may be discrete or real variables, or multidimensional vectors) with an associated similarity measure. For example, to measure the similarity between categorical labels like atom types, the Dirac function $1_{l_i=l_j}$ may be used. In our experiments, we use atom partial charges as atom labels, with a Gaussian kernel $K_{\mathrm{L}}(l_i, l_j) = e^{-\frac{(l_i-l_j)^2}{\lambda}}$. Of course, other similarity measures may be used as well.

Finally, atom labels are used to re-weight the contribution of two atoms x_i and y_j by $K_{\mathrm{L}}(l_i, l_j)$ in (6.2):

$$\text{sup-CK}_L(P_1, P_2) = \max_{R, y_t} \sum_{\substack{x_i \in P_1 \\ y_j \in P_2}} e^{-\frac{(l_i-l_j)^2}{\lambda}} e^{\frac{||x_i-(Ry_j+y_t)||^2}{2\sigma^2}}, \tag{6.4}$$

where parameter λ controls the sensitivity of our measure to atom labels, for example to partial charges. When λ is large, impact of labels is negligible, which corresponds to a purely geometrical approach. When λ is close to zero, only pairs of atoms which have exactly the same partial charge contribute to our measure. In general, the smaller λ, the greater the contribution of the atom labels to the binding pocket similarity measure. Since the function K_{L} does not depend on R and y_t in (6.4), the same optimization procedure can be used to optimize (6.4) or (6.2).

Finally, it is important to notice that the *sup-CK* measure of similarity can be used to compare *any* set of atoms in 3D. While the primary goal of this research is to use it for comparison of binding pockets, we can also use it to compare, e.g., 3D conformations of ligands. This possibility is investigated in the experiments below.

2.2 Related methods

In this section we briefly review some of the existing methods for pocket comparison, which we compare to *sup-CK* in our experiments.

Spherical harmonic decomposition (SHD). Morris et al. [2005] proposed to model pockets by star-shapes built using the SURFNET program. The star-shape representation is defined by a function $f(\theta, \phi)$, representing the distance from the pocket

center to the pocket surface for a given (θ,ϕ). To measure the similarity of binding pockets P_1 and P_2, the corresponding functions f_1 and f_2 are first decomposed into spherical harmonics, and the pocket similarity is then computed as the standard Euclidean metric between vectors of decomposition coefficients. Kahraman et al. [2007] presented three different variants of *SHD*, using only the shapes of binding pockets, the sizes of the binding pockets (keeping only the zero-th order in the spherical harmonics expansion), and their combination. We only present the results of the latter in section 4, because it provided the best performance.

Poisson index (sup-PI). As we already mentioned in Section 2.1, many binding pockets similarity measures are based on pocket alignment with further counting of overlapping atoms. In particular this kind of approach is used in the *Poisson index* model [Davies et al., 2007]. More precisely the *Poisson index* model is based on normalized number of overlapping atoms $PI(P_1, P_2) = \frac{L}{\#P_1+\#P_2-L}$ where L is the number of overlapping atoms, and $\#P_1$ and $\#P_2$ are the respective numbers of atoms in the two pockets. The PI score may be computed for any pocket superposition method.While Davies et al. [2007] used the geometric hashing algorithm, we use in our experiments the superposition made by *sup-CK* method, with further superposition refining to maximize the number of overlapping atoms.

Multibind. Shulman-Peleg et al. [2008] represent pockets by pseudo-atoms labeled with physico-chemical properties. Pockets are aligned using a geometric hashing technique. This algorithm was mainly designed for multiple alignment of binding sites, but it may be used for pairwise alignment of pockets, as was performed in this study.

Other simple methods. We also consider two simple methods based on the comparison of simple binding pockets characteristics. These methods represent each pocket by an ellipsoid constructed on the basis of pocket principal axis. The first method, referred to as *Vol*, estimates the similarity between pockets P_1 and P_2 by the absolute value of the difference between the volumes of their corresponding ellipsoids: $Vol(P_1, P_2) = |Vol(P_1) - Vol(P_2)|$. The second method, called *Princ-Axis*, estimates the similarity score between pockets by $\sum_{i=1}^{3}(\lambda_i^{P_1} - \lambda_i^{P_2})^2$, where $\lambda_i^{P_1}$ and $\lambda_i^{P_2}$ are the lengths of the three principle axis of pockets P_1 and P_2, respectively.

Combination of sup-CK and Vol. Since volume information was found to be important by Kahraman et al. [2007], we also test a linear combination of the *sup-CK* and *Vol* methods, called *sup-CK-Vol*, where the coefficient of linear combination is learned as other model parameters in the double cross validation scheme. This linear combination takes advantage of the Vol method to separate very different pockets like PO4 and NAD, and of the *sup-CK* algorithm to allow finer discrimination.

2.3 Performance criteria

There are various ways to measure the similarity between binding pockets, some of them were discussed in the previous section. To evaluate the quality of a given similarity measure, one may compare it to some "ideal" similarity measure between binding pockets, but the problem is that such measure does not exist. As an example, given two alternative measures SM1 and SM2 applied to two pockets P1 and P2 such that SM1(P1,P2)= 0.3 and SM2(P1,P2)= 0.4, there is no way to decide which one is the best because we do not have any absolute reference. The choice of the optimal measure, thus, may depend on a particular problem of interest. In the context of ligand prediction, the quality of a similarity measure can be evaluated according to its ability to regroup together pockets binding the same ligand, which can be used to predict ligands for previously unseen binding pockets. To evaluate the regrouping quality of the similarity measures, we use two different scores.

AUC score. Kahraman et al. [2007] use the AUC score which is computed as follows. Let us consider a set of pockets $(P_1, \ldots, P_N)$ and a similarity measure SM. To estimate the AUC score of a given pocket P_*, we rank all other pockets according to their similarity to P_*, $SM(P_i, P_*)$ (descending order), and we plot the ROC curve, i.e., the number of pockets binding the same ligand versus the number of pockets binding a different ligand among the top n pockets, when n varies from 0 to N. The ranking quality of SM is measured by the surface of area under the ROC curve, which defines the AUC score. An "ideal" SM function will rank all pockets binding the same ligand as P_* on the top of the list, leading to an AUC score equal to 1.0. On the contrary, if these pockets have random positions in the ranked list, the AUC

score will be equal to 0.5 (worst possible case). Finally, to evaluate the overall AUC score of a method, we consider its mean value over all pockets.

While the AUC score represents an intuitive and natural way to evaluate the quality of similarities measures, in some situations it may fail. Consider the case of a dataset containing two types of pockets L_1 and L_2 (i.e. they bind two different ligands), and a similarity measure that correctly clusters pockets according to their type. If clusters are close to each other (see clusters A and C in Figure 6.1), the AUC score of pockets situated near the border (pockets p_1 and p_2 in Figure 6.1) will be low. The situation becomes even worse, if pockets binding ligand L_1 form several clusters, as shown in Figure 6.1, leading to low AUC scores for almost all pockets binding ligand L_1. This similarity measure will have an overall poor AUC score, although it produces perfect separation of pocket types.This happens, for example, when the database contains proteins that underwent convergent evolution and bind the same ligand under highly different conformations. Therefore, a poor AUC score does not necessarily correspond to a poor pocket separation, and AUC scores may not be suited to evaluate the quality of similarity measures.

Classification error. These remarks lead us to employ another quality score based on classification error. To evaluate the quality of the similarity measure SM we try to predict a ligand (class) for each pocket from that of its neighbors. The smaller the classification error (proportion of bad predictions), the better the similarity measure.

In this work, we use a K nearest neighbors (KNN) classifier. To evaluate the classification error, we applied a leave-one-out double cross validation methodology. Namely, each pocket from the dataset is considered one by one, and all other pockets are used as the training set. Parameters of the model (k — number of neighbors, σ if we consider sup-SM method) are estimated on the training data via cross-validation technique, and the class (i.e. the ligand) of the pocket under consideration is predicted using the training data and the estimated parameters of the model.

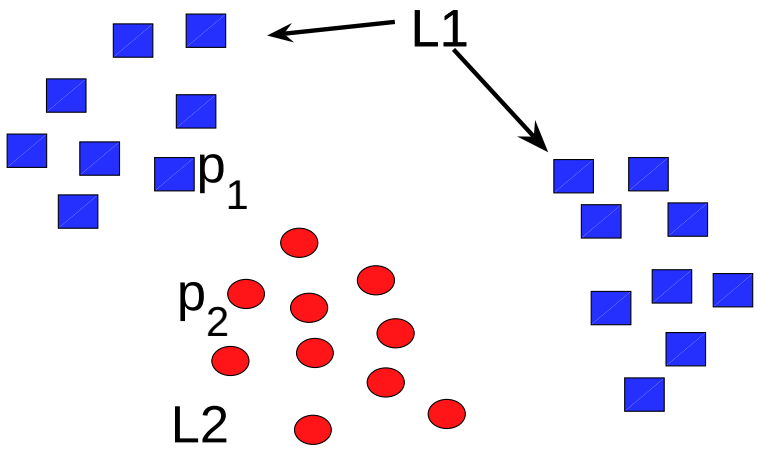


Figure 6.1: AUC score versus classification error as an evaluation of binding pocket similarity measure. Red circles represents pockets fixing ligand L_1, blue squares represents pockets fixing ligand L_2. The AUC score does not reflect the fact of good pocket clusterization, while the classification error does.

2.4 Docking

Since docking programs may also predict ligands, we used the *Fred* [Nicholls, 2005] and *FlexX* [Rarey et al., 1996] programs. We chose these two programs because they are well referenced, and represent different strategies for ligand placement in the binding site. In all docking experiments, the active sites were the same as those used by the *sup-CK* methods. *Fred* performs rigid docking of molecules. Flexibility of ligands is taken into account by using pre-calculated conformers of a molecule. These conformers are ranked according to their estimated interaction energy with the protein, which defines the docking score [Mcgann et al., 2003]. For each pocket, the predicted ligand was the most frequent molecule observed among the K first ranked molecules (K was optimized for each dataset).

FlexX performs flexible docking of molecules by fragmentation and incremental rebuilding inside the binding site. Therefore, only one ligand conformation is required

as input, and the docking results are expected to be independent from that conformation. To predict a ligand for a given pocket, we choose the molecule of best docking score. In all cases, *FlexX* was run using standard parameters, with formal charges, and multiple conformations for rings were computed with Corina [Gasteiger et al., 1996].

To evaluate the performance of docking programs we can use only classification error score. Fred and Flex may be used to predict binding ligands, but they do not measure similarity between binding pockets, so we can not compute the AUC score.

3 Datasets

For all protein structures, the binding pockets were extracted as follows: protein atoms situated at less than RÅ of one of the ligand atoms were selected, where R is considered as a model parameter and is learned in the double cross-validation scheme. In our experiments, in most cases the optimal value of R was equal to 5.3 Å, this distance cutoff is in the range of that above which most interaction energy terms between a protein and a ligand usually become negligible. Finally, pockets are represented by 3D atom clouds with atom labeled by their partial charge, but other labels representing chemical properties such as amino-acid type could be included. Atom partial charges were attributed according to the GROMACS (FFG43a1) force field [Scott et al., 1999].

We consider several benchmark datasets. The first one, referred to as the *Kahraman dataset*, comprises the crystal structures of 100 proteins in complex with one of ten ligands (AMP, ATP, PO4, GLC, FAD, HEM, FMN, EST, AND, NAD). It was proposed by Kahraman et al. [2007] and is described in the Supplementary Materials. We built an extended version of the Kahraman dataset (called *extended Kahraman Dataset* below), also described in the Supplementary Materials, in which we added protein structures in complex with one of the same ten ligands, leading to a total of 972 crystal structures. The added proteins present pairwise sequence identities less or equal to 30%, to avoid potential bias by inclusion of close homologs.

The Kahraman dataset contains only holo protein structures. However, apo structures may differ from holo structures when the latter undergo structural rearrangement upon ligand binding, a phenomenon called induced fit of the protein in order to adjust to the ligand [Bosshard, 2001]. We tested a few examples of predictions for eight apo structures to evaluate the robustness of our method with respect to atom positions variability. We considered 8 apo structures corresponding to proteins able to bind one ligand from the Kahraman database: 1ADE for AMP, 1B8P for NAD, 1E4F for ATP, 1OMP for GLC, 1WS9 for FAD, 2RG7 for HEM, 1X56 for PO4 and 1N05 for FMN. These proteins share less than 30% sequence identity with any of the proteins of the extended Kahraman dataset, and had an holo structure available. The LigASite website [1] was used for this selection. The holo and apo structures of these proteins were superposed, and the coordinates of the ligand in the holo structure were used to extract the pocket in the apo structure.

The Kaharaman dataset comprises ligands of very different sizes and chemical natures. However, the real challenge is to test methods on pockets that bind ligands of similar size. Therefore, we created a third dataset comprising 100 structures of proteins in complex with ten ligands of similar size (ten pockets per ligand). This dataset will be referred to as the *Homogeneous Dataset* (HD), and is described in Supplementary Materials.

4 Results

The methods were tested on two datasets (Section 3 and Supplementary Materials). The performance of all methods is evaluated on the basis of the AUC score and the classification error (Section 2.3). The *sup-CK* method is compared to *sup-PI*, *SHD*, *Vol*, *Princ-Axis* and *MultiBind* algorithms (Section 2.2). Among the pocket extraction methods used in the *SHD* approach, we considered the results corresponding to the Interact Cleft Model, which is similar to our pocket extraction method. Results provided by the docking programs are called *Fred* and *FlexX*.

Pocket representation is subject to extraction noise. To estimate the method

[1] http://www.bigre.ulb.ac.be/Users/benoit/LigASite/

performance on unnoisy systems, algorithms for pockets comparison were also employed to compare ligands (except for the *MultiBind* method which is designed to be employed only on proteins).

4.1 Kahraman Dataset

Results of all methods on the Kahraman Dataset are presented in Table 6.1. According

Table 6.1: Performances for all algorithms evaluated by the mean AUC scores and the mean classification errors (CE), over all pockets. We report only classification error for the Fred and Flex docking programs, because they can not be used to evaluate similarity between binding pockets. Column "Pockets" reports AUC and CE scores based on comparison of binding pockets. Column "Ligands" represents the same thing, but on the basis of ligands, for more explanations see text.

Method	Pockets		Ligands	
	AUC	CE	AUC	CE
sup-CK	0.858±0.14	0.36	0.964±0.006	0.04
sup-CK_L	0.861±0.13	0.27	—	—
sup-CK-Vol	0.889±0.14	0.34	0.985±0.06	0.03
sup-CK_L-Vol	0.895±0.12	0.26	—	—
Vol	0.875±0.14	0.39	0.897±0.13	0.30
Princ-Axis	0.853±0.13	0.35	0.938±0.10	0.16
sup-PI	0.815±0.13	0.42	0.927±0.09	0.05
SHD*	0.770	0.39	0.920	0.07
MultiBind	0.715 ±0.17	0.42	—	—
Fred	—	0.47	—	—
Flexx	—	0.62	—	—

*AUC scores are taken directly from [Kahraman et al., 2007], CE scores are estimated from data provided by authors

to the AUC score, simple methods like *Vol* and *Princ-Axis* give surprisingly good results. The same effect was observed by Kahraman et al. [2007] when they used simple measure based on comparison of pocket sizes. The AUC scores of all the new methods (*sup-CK*, *sup-CK-Vol*, with or without use of partial charges) are higher than those of *ICM*, *MultiBind*, and *sup-PI*, and are in the same range than those of *Vol* and *Princ-Axis*. The best results are obtained by the *sup-CK-Vol* algorithm, which seems to benefit from the association of volume information and of more subtle geometric

details provided by the *sup-CK* algorithm. Another observation, is that information on atom partial charges only leads to modest improvement of the *sup-CK* methods.

To evaluate the classification error, we tried to predict a ligand (a class) for each pocket using a K Nearest Neighbors classifier (see Section 2.3). Note that in a ten class (10 ligands) classification problem, a random classifier would have an error of 0.9, which represents baseline performance for all classifiers.

Table 6.1 shows that methods with higher AUC scores tend to have smaller classification errors, but this correlation is not strict. This indicates that the AUC score is not appropriate to compare similarity measures with respect to the problem of ligand identification, and underlines the interest of the classification approach.

The *sup-CK* and *sup-CK-Vol* algorithms have lower classification errors than other methods, which means that they are well suited to the problem of ligand prediction. Interestingly, atom partial charges information significantly reduces classification errors of both methods, which was not the case for AUC scores. Addition of more information for the description of pockets may improve the quality of ligand prediction. The *SHD* and *MultiBind* methods provide reasonable prediction quality, although they do not perform as well as *sup-CK*. The only difference between the *sup-PI* and *sup-CK* methods is the similarity measure used after superposition. The *sup-PI* method requires to determine the number of overlapping atoms. On the contrary, the *sup-CK* measure is based on a weighted number of atoms having close positions score taking into account, which probably leads to better results.

Docking is now widely used for ligand prediction [Leach et al., 2006], and it is therefore interesting to compare its performances to those of pocket comparison methods. Table 6.1 shows that, on this benchmark, both docking programs do not perform as well as the *sup-CK* method, although *Fred* has better results than *FlexX*. Comparison of docking programs performances is beyond the scope of this work, but it has been widely discussed that relative performances of docking programs strongly depend on the datasets [Warren et al., 2006]. They were here overall modest, but both docking programs better classified pockets associated to large ligands like FAD (flavin-adenine dinucleotide) or FMN (flavin mononucleotide), and poorly those that bind smaller ligands. These results are consistent with the fact that small ligands

make few interactions, leading to low docking scores.

Since *sup-CK* method relies on 3D atom cloud representation of protein pockets, we applied it to compare ligands using their coordinates in the protein-ligand complex structures. We also recall the performances of the *SHD* algorithm for ligands of this dataset. No method reaches an AUC score of 1.0, or perfectly classifies the ligands (i.e. perfectly assign the correct ligand type). This indicates that ligands adopt different conformations in this dataset. However, performances of all algorithms are better for ligands than for pockets. Pockets have to be extracted from the protein structure, which introduces some noise that is absent in the case of ligands. This may explain better results, and represent the best expected performances for each method. In the case of ligand comparison, the best results are obtained with the *sup-CK* algorithms, although those of *SHD* and *sup-PI* are very good. The *Vol* and *Princ-Axis* methods have significantly lower results in terms of ligand classification than other methods, although their AUC scores were in the same range. Similarly, the *SHD* and *sup-PI* AUC scores are close to that of *Princ-Axis*, but they both perform much better in ligand classification than the latter.

Extension of Kahraman dataset.

To evaluate the ability of the *sup-CK* method to improve its performance when trained on a larger dataset, we consider an extension of Kahraman dataset consisting of 972 pockets that bind one of the 10 ligands of the original dataset (see Section 3). Pocket comparison and ligand prediction was performed with the *sup-CK* method including atom partial charges. The mean AUC score and classification error were equal to 0.87 and 0.18. In particular, 79% of the binding pockets of the original Kahraman dataset were correctly classified, compared to 73% on the original dataset (see Table 6.1). The results of the new method improve when trained on a larger dataset, which shows its ability to learn. The quality of predictions might again improve by including more structures available at the PDB.

It is also interesting to study the structure of the dataset according to the metric associated to the *sup-CK* method. We performed kernel principal component analysis [Schölkopf et al., 1999] on the pockets similarity matrix of the *sup-CK* method (this matrix is not positive definite, but we can extract principal components associated to

the largest positive eigenvalues). Figure 6.2(a) represents the projection of 972 binding pockets on the first two principal components. Overall, we observe a clustering of

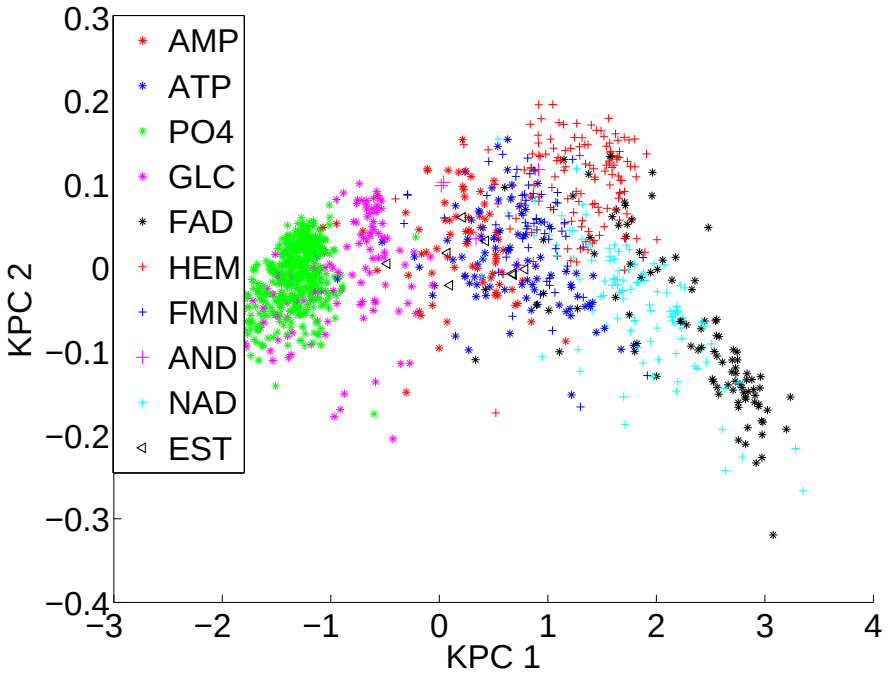


Figure 6.2: Projection of ext-KD on the two first kernel principal components.

binding pockets according to their ligands, which illustrates the good performances of this method for ligand prediction. Looking into more details, we notice that the clusters of pockets that bind ATP, AMP or PO4 overlap. Indeed, proteins that binds ATP usually also bind AMP or PO4, although with different affinities. Furthermore, some pockets (for example pockets that bind glucose GLC or FAD) are found far from their main cluster, or form secondary clusters, which illustrates that pockets having different geometrical characteristics may bind the same ligand. In the classification approach employed here, prediction of a ligand for a given pocket uses the classes of its neighbors, which allows to handle the case of pockets belonging to secondary clusters.

Prediction on apo structures.

The Kahraman dataset includes protein structures in complex with a ligand which was removed, and then predicted in a "leave-one-out" procedure. However, in practice, the relevant problem will be to predict ligand for apo structures. Apo structures may differ from holo structures due to the induced fit phenomenon. Therefore we tested the performance of our method on eight apo structures (Section 3). The ligands for the eight considered apo pockets were predicted by the *sup-CK* algorithm, and the only misclassified pocket was that of 2RG7, a protein which binds HEM.

4.2 Homogeneous dataset (HD)

The Kahraman dataset contains ligands of very different sizes. It is important to test methods on a benchmark containing pockets binding ligands of similar sizes. For this reason, we built the Homogeneous dataset. Table 6.2 presents the performances of different algorithm on this dataset.

Table 6.2: Performances for all algorithms evaluated by the mean AUC scores and the mean classification errors, over all pockets.

Method	Pockets		Ligands	
	AUC	CE	AUC	CE
sup-CK	0.710±0.19	0.47	0.892±0.14	0.12
sup-CK_L	0.752±0.16	0.38	—	—
sup-CK-Vol	0.722±0.18	0.46	0.909±0.17	0.12
sup-CK_L-Vol	0.766±0.17	0.38	—	—
Vol	0.648±0.15	0.89	0.812±0.15	0.54
Princ-Axis	0.650±0.18	0.71	0.830±0.20	0.28
sup-PI	0.702±0.19	0.47	0.880±0.14	0.12
MultiBind	0.69± 0.14	0.48	—	—
Fred	—	0.54	—	—
Flex	—	0.85	—	—

Table 6.2 shows that the performance of all algorithms are lower than on the Kahraman dataset, which illustrates that the Homogeneous dataset is a more difficult benchmark. The *Vol* and *Princ-Axis* display stronger degradation of performances, with AUC scores equal to 0.65, and classification errors of 89% and 71%, respectively. This is due to the fact that the size information is less discriminative on this dataset.

In terms of AUC scores, the best performance is obtained by the *sup-CK* and *sup-CK-Vol* algorithms, but volume information only provides a slight improvement of 1%, compared to 3% on the Kahraman dataset. On the contrary, partial charges information leads to a significant improvement of 4% for the *sup-CK* and *sup-CK-Vol* algorithms. This shows that addition of physico-chemical information is critical for discriminating pockets of similar sizes. In terms of classification error, volume information is useless, but the use of information on partial charge leads to significant improvement of 9%.

The same conclusions also hold for ligands comparison: performances are lower than on the Kaharaman dataset, for all methods, and degradation of the classification errors is much stronger for the *Vol* and *Princ-Axis* methods. On this dataset, the docking programs did not perform as well as methods based on pocket comparison in terms of classification errors.

5 Discussion

An important characteristic of the *sup-CK* algorithm is its ability to adapt to the pocket variability. Parameter σ of the *sup-CK* method controls the sensitivity of the similarity measure to atom relative displacements. The larger the variability of pockets binding the same ligand, the greater should be the value of σ. Figure 6.3a shows how the AUC score and classification error vary with σ on the Homogeneous dataset. In both cases, the optimum is reached when σ is equal to one. Note that, in our experiments (section 4), we did not use the same value of σ estimated from all pockets. For each pocket, the optimal value was estimated on the basis of the 99 training pockets to avoid overfitting to the data. However, we observed that, in most cases (90%), $\sigma = 1$ was chosen. Similarly, when information on atom partial charges is used, parameter λ (6.4) conditions the sensitivity of the method to relative values of atom charges. Figures 6.3b and 6.3c present the variation of AUC scores and classification error as functions of σ and λ. We observe that for the AUC score, the optimum is reached when σ equals to 2 and λ equals to 0.25, while for the classification error optimal σ is equal to 4.

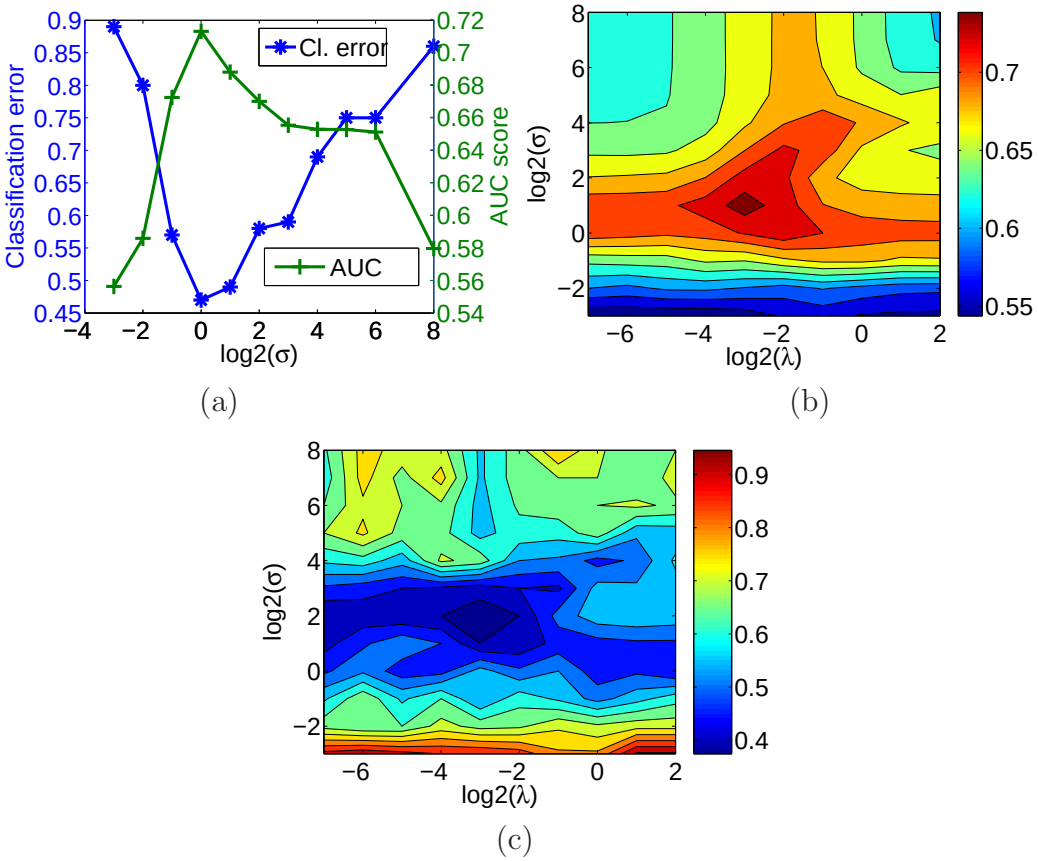


Figure 6.3: Homogeneous database. (a) AUC score and prediction error as functions of σ in the sup-CK method (pure geometrical version,$\lambda = \infty$), (b) AUC score and (c) classification error as functions of σ and λ when information on atom partial charges is used.

Figures 6.4b and 6.4c illustrate the optimal alignment found for two ATP binding pockets. While this alignment was estimated on the basis of pocket atom coordinates, the bound ligands are found well aligned, which suggests a good quality of pocket alignment. Note, that *sup-CK* does not try to superpose individual atoms, but rather superposes atom sets.

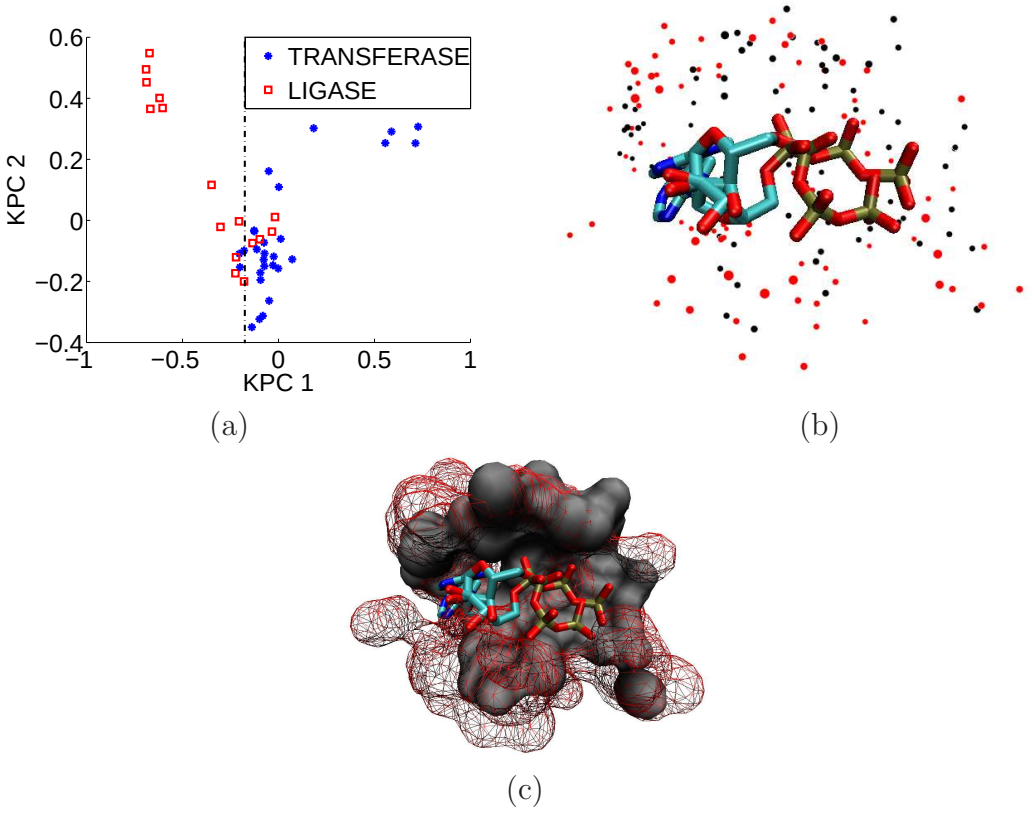


Figure 6.4: (a) Projection of ATP binding pockets on the two first kernel principal components of *sup-CK*. (b,c) Alignment of two ATP pockets made by *sup-CK*, atoms of different pockets are represented by black and red points in (b) and by black and red surfaces in (c), two ATP ligands are traced in licorice.

The running time of the *sup-CK* method depends on the value of stopping criterion used in the gradient ascent method and on the number of atoms. In our experiments, the algorithm running time varied between 0.2 and 1.3 seconds (2.5 GHz CPU) per pocket pair. This running time is already quite reasonable to process large protein

databanks, however a pre-filtering on the basis of simple pocket descriptors (like volume or size) may be quite useful in the further acceleration of the *sup-CK* method. We defined pockets as the set of all protein atoms within 5Å of a bound ligand. Similar approaches were used by Kahraman et al. [2007] (Interacted Cleft Model), and similar pockets may also be retrieved by methods like *Q-SiteFinder* [Laurie and Jackson, 2005] without any information on ligand coordinates.

In our experiments, docking programs (*FlexX* and *Fred*) did not perform as well for ligand prediction as most methods based on pockets similarity measure. Docking programs have many parameters that can be tuned to particular protein-ligand systems [Andersson et al., 2007]. Fine preparation of the active site, such as assignment of amino acid protonation states, is also critical. Such tuning for each pocket is hardly automatized in large scale datasets (up to almost 1000 proteins in this study), and therefore, the performance of docking programs is underestimated.

An important topic is the relation between methods for binding pockets comparison and algorithms in field of computer vision for comparison of 3D shapes. A complete review of 3D shape comparison methods is out of scope of this article, and interested readers may consult [Iyer et al., 2005] for a detailed review. Interestingly, most of existing methods for binding pocket comparison have an analogue in the domain of computer vision. For example, methods based on real spherical harmonic expansion used in [Morris et al., 2005] for binding pocket comparison are also discussed by Papadakis et al. [2007]; Saupe and Vranic [2001] in the context of general 3D shape matching. Principles used in another popular method for matching and comparison of 3D forms, called Iterative Closest Point algorithm [Zhang, 1992], and its variants are used in *Poisson index* and *MultiBind* algorithms. Examples of approaches based on graph representation of 3D forms and graph matching methods may be found in [Weskamp et al., 2007] for binding pockets comparison, as well as in [Biasotti et al., 2004] for 3D shapes comparison. Nevertheless, binding pockets are not continuous shapes but discrete clouds of points. They can be transformed into 3D shapes [Morris et al., 2005; Kahraman et al., 2007], but this transformation may be a source of noise. Moreover, a similarity measure between binding pockets should be rotationally and translationally invariant, which is not always the case in computer

vision methods. However, we believe that the adaptation of appropriate methods may be very fruitful for the recognition of binding pockets.

The prediction of protein ligands is related to the problem of predicting the protein molecular function. We analyzed the repartition of the ATP binding pockets generated by this similarity measure on the extended Kahraman dataset. Figure 6.4a presents the projection of ATP pockets annotated as transferases or ligases, on the first two principal components of the *sup-CK* similarity matrix. We observed that these two families of enzymes are essentially separated. Although these are very preliminary results, they show that *sup-CK* method may be useful in the prediction of protein molecular functions.

The *sup-CK* algorithm showed a good performance in ligand prediction for apo structures. This is an important preliminary result, in order to apply the method to real case studies, or to proteins with no known experimental structure but for which a homology model can be constructed [Launay and Simonson, 2008].

Bibliography

R. Aebersold and M. Mann. Mass spectrometry-based proteomics. *Nature*, 422(6928): 198–207, Mar 2003.

E.L. Allgower and K.Georg. *Numerical continuation methods*. Springer, 1990. ISBN 3-540-12760-7.

H.A. Almohamad and S.O. Duffuaa. A linear programming approach for the weighted graph matching problem. *IEEE Trans. Pattern Anal. Mach. Intell.*, 15(5):522–525, May 1993.

R. Ambauen, S. Fischer, and H. Bunke. Graph edit distance with node splitting and merging, and its application to diatom idenfication. In *GbRPR*, pages 95–106, 2003.

C. D. Andersson, E. Thysell, A. Lindström, M. Bylesjö, F. Raubacher, and A. Linusson. A multivariate approach to investigate docking parameters' effects on docking performance. *J. Chem. Inform. Model.*, 47(4):1673–1687, 2007.

K. M. Anstreicher and N. W. Brixius. A new bound for the quadratic assignment problem based on convex quadratic programming. *Math. Program.*, 89(3):341–357, 2001.

D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. Concorde tsp solver. `http://www.tsp.gatech.edu/concorde.html`, 2005.

D. L. Applegate, R. E. Bixby, V. Chvatal, and W. J. Cook. *The Traveling Salesman Problem: A Computational Study (Princeton Series in Applied Mathematics)*. Princeton University Press, January 2007. ISBN 0691129932.

S. Bandyopadhyay, R. Sharan, and T. Ideker. Systematic identification of functional orthologs based on protein network comparison. *Genome Res.*, 16(3):428–435, Mar 2006.

S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24(4):509–522, 2002.

J. Berg and M. Lässig. Cross-species analysis of biological networks by bayesian alignment. *Proc. Natl. Acad. Sci. USA*, 103(29):10967–10972, Jul 2006.

S. Berretti, A. Del Bimbo, and P. Pala. A graph edit distance based on node merging. In *Proc. of ACM International Conference on Image and Video Retrieval (CIVR)*, pages 464–472, Dublin, Ireland, July 2004.

D. Bertsekas. *Nonlinear programming*. Athena Scientific, 1999.

S. Biasotti, S. Marini, M. Mortara, G. Patane, M. Spagnuolo, and B. Falcidieno. 3d shape matching through topological structures. In *Discrete Geometry for Computer Imagery*, pages 194–203. Springer Berlin / Heidelberg, 2004.

N. L. Biggs, E.K. Lloyd, and R. J. Wilson. *Graph theory 1736-1936*. Oxford University Press, 1976.

A. Blake and A. Zisserman. *Visual Reconstruction*. MIT Press, 1987.

J. A. Bondy and U. S. R. Murty. *Graph theory with applications*. Macmillan Press Ltd., 1976.

F. L. Bookstein. Principal warps: thin-plate splines and the decomposition of deformations. *IEEE T. Pattern. Anal.*, 11(6):567–585, 1989.

J. M. Borwein and A. S. Lewis. *Convex Analysis and Nonlinear Optimization*. Springer-Verlag, New York, 2000.

H. R. Bosshard. Molecular recognition by induced fit: how fit is the concept? *News Physiol Sci*, 16:171–173, Aug 2001.

S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, USA, 2004. ISBN 0521833787.

K. Brein, M. Remm, and E. Sonnhammer. Inparanoid: a comprehensive database of eukaryothic orthologs. *Nucleic Acids Res.*, 33, 2005.

H. Bunke. Inexact graph matching for structural pattern recognition. *Pattern Recogn. Lett.*, 1(4):245–253, May 1983. ISSN 01678655.

L. Buriol, P. M. França, and P. Moscato. A new memetic algorithm for the asymmetric traveling salesman problem. *Journal of Heuristics*, 10(5):483–506, 2004. ISSN 1381-1231.

T. Caelli and S. Kosinov. An eigenspace projection clustering method for inexact graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 26(4):515–519, April 2004.

C. Callison-Burch, P. Koehn, C. Monz, J. Schroeder, and C. S. Fordyce, editors. *Proceedings of the Third Workshop on SMT*. ACL, Columbus, Ohio, June 2008.

M. Carcassoni and E. Hancock. Spectral correspondence for point pattern matching. *Pattern Recogn.*, 36(1):193–204, January 2003. ISSN 00313203.

A. Cayley. On the theory of the analytical forms called threes. *Philos. Mag.*, 37(18): 374–378, 1859.

A. Cayley. On the mathematical theory of isomers. *Philos. Mag.*, 10(47):444–446, 1874.

A. Cayley. On the theory of the analytical forms called threes, with application to the theory of chemical combinations. *Rep. Brit. Assoc. Sci.*, 4(45):257–305, 1875.

A. Cayley. On the number of univalent radicals $c_n h_{2n+1}$. *Philos. Mag.*, 18(3):34–35, 1877.

E. Cela. Qaudratuc assignment problem library, 2007. URL www.opt.math.tu-graz.ac.at/qaplib/.

F. R. K. Chung. *Spectral graph theory*, volume 92 of *CBMS Regional Conference Series*. American Mathematical Society, Providence, 1997.

W. K. Clifford. Binary forms of alternate variables. *Proc. London Math. Soc.*, 10(9): 277–286, 1878a.

W. K. Clifford. Note on quantics of alternate numbers, used as a means for determining the invariants and covariants of quantics in general. *Proc. London Math. Soc.*, 10(9):258–265, 1878b.

D. Conte, P. Foggia, C. Sansone, and M. Vento. Thirty years of graph matching in pattern recognition. *Int. J. Pattern. Recogn. Artif. Intell.*, 18(3):265–298, 2004.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An efficient algorithm for the inexact matching of arg graphs using a contextual transformational model. *Pattern Recognition, International Conference on*, 3:180, 1996. ISSN 1051-4651.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. Performance evaluation of the vf graph matching algorithm. In *ICIAP '99: Proceedings of the 10th International Conference on Image Analysis and Processing*, page 1172, Washington, DC, USA, 1999. IEEE Computer Society. ISBN 0-7695-0040-4.

L. P. Cordella, P. Foggia, C. Sansone, and M. Vento. An improved algorithm for matching large graphs. In *In: 3rd IAPR-TC15 Workshop on Graph-based Representations in Pattern Recognition, Cuen*, pages 149–159, 2001.

T. Cour, P. Srinivasan, and J. Shi. Balanced graph matching. In *Advanced in Neural Information Processing Systems*, 2006.

J.R. Davies, R.M. Jackson, K.V. Mardia, and C.C. Taylor. The poisson index: a new probabilistic model for protein ligand binding site similarity. *Bioinformatics*, 23 (22):3001–3008, Nov 2007.

R. Diestel. *Graph theory*. Springer-Verlag, 2000.

R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological Sequence Analysis: Probabilistic Models of Proteins and Nucleic Acids*. Cambridge University Press, 1998.

S. Fields and O. Song. A novel genetic system to detect protein-protein interactions. *Nature*, 340(6230):245–246, Jul 1989.

A. Filatov, A. Gitis, and I. Kil. Graph-based handwritten digit string recognition. In *ICDAR '95: Proceedings of the Third International Conference on Document Analysis and Recognition (Volume 2)*, page 845, Washington, DC, USA, 1995. IEEE Computer Society. ISBN 0-8186-7128-9.

J. Flannick, A. Novak, B.S. Srinivasan, H.H. McAdams, and S. Batzoglou. Graemlin: general and robust alignment of multiple large interaction networks. *Genome Res.*, 16(9):1169–1181, Sep 2006.

S. Fortin. The graph isomorphism problem. Technical report, MIT, 1996.

M. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

M. R. Garey and D. S. Johnson. *Computer and intractability: A guide to the theory of NP-completeness*. San Francisco, CA: W. H. Freeman, 1979.

T. Gärtner, P.A. Flach, A. Kowalczyk, and A.J. Smola. Multi-Instance Kernels. In C. Sammut and A. Hoffmann, editors, *Proceedings of the Nineteenth International Conference on Machine Learning*, pages 179–186. Morgan Kaufmann, 2002.

J. Gasteiger, J. Sadowski, J. Schuur, P. Selzer, L. Steinhauer, and V. Steinhauer. Chemical information in 3d space. *J. Chem. Inform. Comput. Sci.*, 36(5):1030–1037, 1996.

G. Gati. Further annotated bibliography on the isomorphism disease. *J. Graph Theor.*, 3:95–109, 1979. ISSN 0364-9024.

U. Germann, M. Jahr, K. Knight, and D. Marcu. Fast decoding and optimal decoding for machine translation. In *In Proceedings of ACL 39*, pages 228–235, 2001.

F. Glaser, R. J. Morris, R. J. Najmanovich, R. A. Laskowski, and J. M. Thornton. A method for localizing ligand binding pockets in protein structures. *Proteins*, 62 (2):479–488, February 2006. ISSN 1097-0134.

C. Godsil and G. Royle. *Algebraic graph theory*. Springer-Verlag, 2000.

N.D. Gold and R.M. Jackson. Sitesbase: a database for structure-based protein-ligand binding site comparisons. *Nucleic Acids Res.*, 34:D231–D234, Jan 2006.

S. Gold and A. Rangarajan. A graduated assignment algorithm for graph matching. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18(4):377–388, April 1996.

G. H. Golub and C. F. Van Loan. *Matrix computations (3rd ed.)*. Johns Hopkins University Press, Baltimore, MD, USA, 1996. ISBN 0-8018-5414-8.

L. Gotusso and A. T. Santolini. A fortran iv quasi decision algorithm for the p-equivalence of two matrices. *Calcolo*, 5:17–35, 1957.

G. Gutin. Travelling salesman and related problems. In *Handbook of Graph Theory*, 2003.

G. Gutin, D. Karapetyan, and N. Krasnogor. Memetic algorithm for the generalized asymmetric traveling salesman problem. In *NICSO 2007*, pages 199–210. Springer Berlin, 2008.

T. Gärtner, K. Driessens, and J. Ramon. Exponential and geometric kernels for graphs. In *Mach. Learn.*, pages 146–163. Springer, 2002.

D. Haussler. Convolution Kernels on Discrete Structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz, 1999.

H. Hoang and P. Koehn. Design of the Moses decoder for statistical machine translation. In *ACL 2008 Software workshop*, pages 58–65, Columbus, Ohio, June 2008. ACL.

N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. Three-dimensional shape searching: state-of-the-art review and future trends. *Computer-Aided Design*, 37(5):509–530, April 2005.

D.S. Johnson, G. Gutin, L.A. McGeoch, A. Yeo, W. Zhang, and A. Zverovich. Experimental analysis of heuristics for the atsp. In *The Travelling Salesman Problem and Its Variations*, pages 445–487, 2002.

M. Jordan, editor. *Learning in Graphical Models*. The MIT Press, 2001.

A. Kahraman, R. J. Morris, R. A. Laskowski, and J. M. Thornton. Shape variation in protein binding pockets and their ligands. *J. Mol. Biol.*, 368(1):283–301, Apr 2007.

A. C. Kam and G. E. Kopec. Document image decoding by heuristic search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 18:945–950, 1996.

H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized Kernels between Labeled Graphs. In T. Faucett and N. Mishra, editors, *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328, New York, NY, USA, 2003. AAAI Press.

B.P. Kelley, R. Sharan, R.M. Karp, T. Sittler, D.E. Root, B.R. Stockwell, and T. Ideker. Conserved pathways within bacteria and yeast as revealed by global protein network alignment. *Proc. Natl. Acad. Sci. USA*, 100(20):11394–11399, Sep 2003.

B.P. Kelley, B. Yuan, F. Lewitter, R. Sharan, B.R. Stockwell, and T. Ideker. PathBLAST: a tool for alignment of protein interaction networks. *Nucleic Acids Res.*, 32(Web Server issue):W83–W88, Jul 2004.

Y. Keselman, A. Shokoufandeh, M. F. Demirci, and S. Dickinson. Many-to-many graph matching via metric embedding. In *CVPR*, pages 850–857, 2003.

K. Knight. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25:607–615, 1999.

P. Koehn, F. J. Och, and D. Marcu. Statistical phrase-based translation. In *NAACL 2003*, pages 48–54, Morristown, NJ, USA, 2003. Association for Computational Linguistics.

R. Kondor and K. M. Borgwardt. The skew spectrum of graphs. In *ICML '08: Proceedings of the 25th international conference on Machine learning*, pages 496–503, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-205-4.

R. Kondor and T. Jebara. A kernel between sets of vectors. In *In International Conference on Machine Learning (ICML)*, 2003.

R. Kondor, N. Shervashidze, and K. M. Borgwardt. The graphlet spectrum. In *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*, pages 529–536, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1.

M. Koyutürk, Y. Kim, U. Topkara, S. Subramaniam, W. Szpankowski, and A. Grama. Pairwise alignment of protein interaction networks. *J. Comput. Biol.*, 13(2):182–199, Mar 2006.

H. W. Kuhn. The Hungarian method for the assignment problem. *Naval Research*, 2:83–97, 1955.

G. Launay and T. Simonson. Homology modelling of protein-protein complexes: a simple method and its possibilities and limitations. *BMC Bioinformatics*, 9:427, 2008.

A. T. R. Laurie and R. M. Jackson. Q-sitefinder: an energy-based method for the prediction of protein–ligand binding sites. *Bioinformatics*, 21(9):1908–1916, 2005. ISSN 1367-4803.

A. R. Leach, B. K. Shoichet, and C. E. Peishoff. Prediction of protein-ligand interactions. docking and scoring: successes and gaps. *J. Med. Chem.*, 49(20):5851–5855, Oct 2006.

R. S. T. Lee and J. N. K. Liu. An oscillatory elastic graph matching model for recognition of offline handwritten chinese characters. In *KES*, pages 284–287, 1999.

M. Leordeanu and M. Hebert. A spectral technique for correspondence problems using pairwise constraints. In *International Conference of Computer Vision (ICCV)*, volume 2, pages 1482 – 1489, October 2005.

M. Leordeanu, M. Hebert, and R. Sukthankar. Beyond local appearance: Category recognition from pairwise interactions of simple features. In *Proceedings of CVPR*, June 2007.

A. Lopez. Statistical machine translation. *ACM Comput. Surv.*, 40(3):1–49, 2008. ISSN 0360-0300.

A. C. Lunn and J. K. Senior. Isomerism and configuration. *J. Phys. Chem.*, 33: 1027–1079, 1929.

B. Luo and E. R. Hancock. Alignment and correspondence using singular value decomposition. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pages 226–235, London, UK, 2000. Springer-Verlag. ISBN 3-540-67946-4.

P. Mahé, N. Ueda, T. Akutsu, J.-L. Perret, and J.-P. Vert. Extensions of marginalized graph kernels. In R. Greiner and D. Schuurmans, editors, *Proceedings of the Twenty-First International Conference on Machine Learning (ICML 2004)*, pages 552–559. ACM Press, 2004.

M. R. Mcgann, H. R. Almond, A. Nicholls, A. J. Grant, and F. K. Brown. Gaussian docking functions. *Biopolymers*, 68(1):76–90, 2003.

L. F. McGinnis. Implementation and testing of a primal-dual algorithm for the assignment problem. *Operations Research*, 31(2):277–291, 1983.

J.W. Milnor. *Topology from the Differentiable Viewpoint*. Univ. Press of Virginia, 1969. ISBN 978-0-691-04833-8.

R. J. Morris, R.J. Najmanovich, A. Kahraman, and J.M. Thornton. Real spherical harmonic expansion coefficients as 3d shape descriptors for protein binding pocket and ligand comparisons. *Bioinformatics*, 21(10):2347–2355, May 2005.

R. Najmanovich, N. Kurbatova, and J. Thornton. Detection of 3d atomic similarities and their use in the discrimination of small molecule protein-binding sites. *Bioinformatics*, 24(16):i105–i111, Aug 2008.

M. Neuhaus and H. Bunke. *Bridging the Gap Between Graph Edit Distance and Kernel Machines*. World Scientific, September 2007.

M. Neuhaus, K. Riesen, and H. Bunke. Fast suboptimal algorithms for the computation of graph edit distance. In Dit-Yan Yeung, James T. Kwok, Ana L. N. Fred, Fabio Roli, and Dick de Ridder, editors, *SSPR/SPR*, volume 4109 of *Lecture Notes in Computer Science*, pages 163–172. Springer, 2006. ISBN 3-540-37236-9.

M. E. J. Newman, S. H. Strogatz, and D. J. Watts. Random graphs with arbitrary degree distributions and their applications. *Phys. Rev. E*, 64:26118, 2001.

A. Nicholls. Oechem, version 1.3.4, openeye scientific software. website, 2005.

C. Noon and J.C. Bean. An efficient transformation of the generalized traveling salesman problem. *INFOR*, pages 39–44, 1993.

P. Papadakis, I. Pratikakis, S. Perantonis, and T. Theoharis. Efficient 3d shape matching and retrieval using a concrete radialized spherical projection representation. *Pattern Recogn.*, 40(9):2437–2452, 2007. ISSN 0031-3203.

K. Papineni, S. Roukos, T. Ward, and W. J. Zhu. BLEU: a Method for Automatic Evaluation of Machine Translation. *IBM Research Report*, RC22176, 2001.

P.M. Pardalos, L. S. Pitsoulis, and M. G. C. Resende. A parallel grasp implementation for the quadratic assignment problem. In *Parallel Algorithms for Irregularly Structured Problems*, pages 111–130. Kluwer Academic Publishers, 1995.

G. Pólya. Algebraische berechnung der anzhal der isomeren einiger organischer verbindungen. *Z. Kristal.*, 93:415–443, 1936.

K. Popat, D. H. Greene, J. K. Romberg, and D. S. Bloomberg. Adding linguistic constraints to document image decoding: Comparing the iterated complete path and stack algorithms, 2001.

A. Rangarajan and E. Mjolsness. A lagrangian relaxation network for graph matching. In *IEEE Trans. Neural Networks*, pages 4629–4634. IEEE Press, 1996.

M. Rarey, B. Kramer, T. Lengauer, and G. Klebe. A fast flexible docking method using an incremental construction algorithm. *J. Mol. Biol.*, 261(3):470–489, Aug 1996.

L. C. Ray and R. A. Kirsch. Finding chemical records by digital computers. *Science*, 126:814, 1957.

R. C. Read and D. G. Corneil. The graph isomorphism disease. *J. Graph Theor.*, 1 (4):339–363, 1977.

J. H. Redfield. The theory of group-reduced distributions. *Amer. J. Math.*, 49: 433–455, 1927.

M. Remm, C.E. Storm, and E.L. Sonnhammer. Automatic clustering of orthologs and in-paralogs from pairwise species comparisons. *J. Mol. Biol.*, 314(5):1041–1052, Dec 2001.

R. Rockafeller. *Convex Analysis*. Princeton Univ. Press, 1970.

T. Saito, H. Yamada, and K. Yamamoto. On the data base etl9b of handprinted characters in jis chinese characters and its analysis. *IEICE Trans*, 68, 1985.

D. Saupe and D. V. Vranic. 3d model retrieval with spherical harmonics and moments. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 392–397, London, UK, 2001. Springer-Verlag. ISBN 3-540-42596-9.

C. Schalon, J-S. Surgand, E. Kellenberger, and D. Rognan. A simple and fuzzy method to align and compare druggable ligand-binding sites. *Proteins*, 71(4):1755–1778, Jun 2008.

C. Schellewald and C. Schnorr. Probabilistic subgraph matching based on convex relaxation. In *EMMCVPR05*, pages 171–186, 2005.

C. Schellewald, S. Roth, and C. Schnörr. Evaluation of convex optimization techniques for the weighted graph-matching problem in computer vision. In *Proceedings of the 23rd DAGM-Symposium on Pattern Recognition*, pages 361–368, London, UK, 2001. Springer-Verlag. ISBN 3-540-42596-9.

D. C. Schmidt and L. E. Druffel. A fast backtracking algorithm to test directed graphs for isomorphism using distance matrices. *J. ACM*, 23(3):433–445, 1976. ISSN 0004-5411.

B. Schölkopf, A.J. Smola, and K.-R. Müller. Kernel principal component analysis. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, pages 327–352. MIT Press, 1999.

B. Schölkopf, K. Tsuda, and J.-P. Vert. *Kernel Methods in Computational Biology*. MIT Press, The MIT Press, Cambridge, Massachussetts, 2004.

W. R. P. Scott, I. G. Tironi, A. E. Mark, S. R. Billeter, J. F., A. E. Torda, T. Huber, and P. Kruger. The gromos biomolecular simulation program package. *J. Phys. Chem. A*, 103:3596–3607, 1999.

L. S. Shapiro and M. Brady. Feature-based correspondence: an eigenvector approach. *Image Vision Comput.*, 10(5):283–288, 1992.

R. Sharan, S. Suthram, R.M. Kelley, T. Kuhn, S. McCuine, P. Uetz, T. Sittler, R.M. Karp, and T. Ideker. Conserved patterns of protein interaction in multiple species. *Proc. Natl. Acad. Sci. USA*, 102(6):1974–1979, Feb 2005.

A. Shulman-Peleg, M. Shatsky, R. Nussinov, and H. J. J. Wolfson. Multibind and mappis: webservers for multiple alignment of protein 3d-binding sites and their interactions. *Nucleic Acids Res.*, 36:260–264, May 2008. ISSN 1362-4962.

R. Singh, J. Xu, and B. Berger. Global alignment of multiple protein interaction networks with application to functional orthology detection. *Proc. Natl. Acad. Sci. USA*, 105(35):12763–12768, Sep 2008.

K. Sjölander. Phylogenomic inference of protein molecular function: advances and challenges. *Bioinformatics*, 20(2):170–179, Jan 2004.

E. H. Sussenguth. A graph-theoretic algorithm for matching chemical structures. *J. Chem. Doc.*, 5(1):36–43, 1963.

S. Suthram, T. Sittler, and T. Ideker. The plasmodium protein network diverges from those of other eukaryotes. *Nature*, 438(7064):108–112, Nov 2005.

J. J. Sylvester. Chemistry and algebra. *Nature*, 17(432), 1878.

W. R. Taylor. Protein structure comparison using bipartite graph matching and its application to protein structure classification. *Mol. Cell. Proteomics*, 1(4):334–339, April 2002. ISSN 1535-9476.

C. Tillmann. Efficient Dynamic Programming Search Algorithms For Phrase-Based SMT. In *Workshop On Computationally Hard Problems And Joint Inference In Speech And Language Processing*, 2006.

C. Tillmann and H. Ney. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Comput. Linguist.*, 29(1):97–133, 2003. ISSN 0891-2017.

W.H. Tsai and K.S. Fu. Error-correcting isomorphisms of attributed relational graphs for pattern analysis. *SMC*, 9(12):757–768, December 1979.

J. R. Ullmann. An algorithm for subgraph isomorphism. *J. ACM*, 23(1):31–42, 1976. ISSN 0004-5411.

S. Umeyama. An eigendecomposition approach to weighted graph matching problems. *IEEE Trans. Pattern Anal. Mach. Intell.*, 10(5):695–703, Sept. 1988.

S. V. N. Vishwanathan, K. M. Borgwardt, R. I. Kondor, and N. N. Schraudolph. Graph kernels. *CoRR*, abs/0807.0093, 2008.

A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, 13(2):260–269, 1973.

T. Walter, J.-C. Klein, P. Massin, and A. Erignay. Detection of the median axis of vessels in retinal images. *European Journal of Ophthalmology*, 13(2), 2003.

H. F. Wang and E. R. Hancock. Correspondence matching using kernel principal components analysis and label consistency constraints. *Pattern Recogn.*, 39(6): 1012–1025, 2006. ISSN 0031-3203.

Y. Wang, F. Makedon, and J. Ford. A bipartite graph matching framework for finding correspondences between structural elements in two proteins. In *Proceedings of the 26th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, 2004.

G.L. Warren, C.W. Andrews, A.M. Capelli, B. Clarke, J. LaLonde, M.H. Lambert, M. Lindvall, N. Nevins, S.F. Semus, S. Senger, G. Tedesco, I.D. Wall, J.M. Woolven, C.E. Peishoff, and M.S. Head. A critical assessment of docking programs and scoring functions. *J. Med. Chem.*, 49(20):5912–5931, Oct 2006.

N. Weskamp, E. Hullermeier, D. Kuhn, and G. Klebe. Multiple graph alignment for the structural analysis of protein active sites. *IEEE/ACM Trans. Comput. Biol. Bioinformatics*, 4(2):310–320, 2007. ISSN 1545-5963.

H. Whitney. Non-separable and planar graphs. *Proc. Natl. Acad. Sci.*, 93:415–443, 1930.

H. Whitney. Congruent graphs and the connectivity of graphs. *Amer. J. Math.*, 54: 150–168, 1932.

Wikipedia. Travelling Salesman Problem — Wikipedia, The Free Encyclopedia, 2009. [Online; accessed 5-May-2009].

P. Willett. From chemical documentation to chemoinformatics: 50 years of chemical information science. *J. Inf. Sci.*, 34(4):477–499, 2008. ISSN 0165-5515.

P. Willett, V. Winterman, and D. Bawden. Implementation of nearest-neighbor searching in an online chemical structure search system. *J. Chem. Inform. Comput. Sci.*, 26(1):36–41, 1986.

L. Xu and I. King. A pca approach for fast retrieval of structural patterns in attributed graphs. In *Humboldt University Berlin*, 1994.

N. Yosef, R. Sharan, and W. S. Noble. Improved network-based identification of protein orthologs. *Bioinformatics*, 24(16):i200–i206, Aug 2008.

M. Zaslavskiy, F. Bach, and J. P. Vert. GRAPHM: Graph matching package, 2008a. Available at `http://cbio.ensmp.fr/graphm`.

M. Zaslavskiy, F. Bach, and J. P. Vert. A path following algorithm for graph matching. In A. Elmoataz, O. Lezoray, F. Nouboud, and D. Mammass, editors, *Image and Signal Processing, Proceedings of the 3rd International Conference, ICISP 2008*, volume 5099 of *LNCS*, pages 329–337. Springer Berlin / Heidelberg, 2008b.

M. Zaslavskiy, F. Bach, and J.-P. Vert. A path following algorithm for the graph matching problem. Technical Report 00232851, HAL, 2008c. To appear in IEEE Trans. Pattern Anal. Mach. Intell.

M. Zaslavskiy, M. Dymetman, and N. Cancedda. Phrase-based statistical machine translation as a traveling salesman problem. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 333–341, Suntec, Singapore, August 2009. Association for Computational Linguistics.

M. Zaslavskiyi, F. Bach, and J-P. Vert. Global alignment of protein-protein interaction networks by graph matching methods. *Bioinformatics*, 25(12), 2009.

Z. Zhang. Iterative point matching for registration of free-form curves. Technical report, Institut National de Recherche en Informatique et en Automatique (INRIA), 1992.